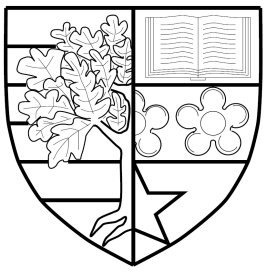
MULTI-FIDELITY DEEP RESIDUAL RECURRENT NEURAL NETWORKS FOR UNCERTAINTY QUANTIFICATION

by

J.Nagoor kani



Submitted for the degree of
Doctor of Philosophy

Department of Petroleum Engineering
School of Energy, Geoscience, Infrastructure and Society
Heriot-Watt University

October 2018

Abstract

Effective propagation of uncertainty through a nonlinear dynamical system is an essential task for a number of engineering applications. One viable probabilistic approach to propagate the uncertainty from the high dimensional random inputs to the high-fidelity model outputs is Monte Carlo method. However, Monte Carlo method requires a substantial number of computationally expensive high-fidelity simulations to converge their computed estimations towards the desired statistics. Hence, performing Monte Carlo high-fidelity simulations becomes computationally prohibitive for large-scale realistic problems. Multi-fidelity approaches provide a general framework for combining a hierarchy of computationally cheap low-fidelity models to accelerate the Monte Carlo estimation of the high-fidelity model output. The objective of this thesis is to derive computationally efficient low-fidelity models and an effective multi-fidelity framework to accelerate the Monte Carlo method that uses a single high-fidelity model only.

In this thesis, a physics aware recurrent neural network (RNN) called deep residual recurrent neural network (DR-RNN) is developed as an efficient low-fidelity model for nonlinear dynamical systems. The information hidden in the mathematical model representing the nonlinear dynamical system is exploited to construct the DR-RNN architecture. The developed DR-RNN is inspired by the iterative steps of line search methods in finding the residual minimiser of numerically discretized differential equations. More specifically, the stacked layers of the DR-RNN architecture is formulated to act collectively as an iterative scheme. The dynamics of DR-RNN is explicit in time with remarkable convergence and stability properties for a large time step that violates numerical stability condition. Numerical examples demonstrate that DR-RNN can effectively emulate the high-fidelity model of nonlinear physical systems with a significantly lower number of parameters in comparison to standard RNN architectures. Further, DR-RNN is combined with Proper Orthogonal Decomposition (POD) for model reduction of time dependent partial differential equations. The numerical results show the proposed DR-RNN as an explicit and

stable reduced order technique. The numerical results also show significant gains in accuracy by increasing the depth of proposed DR-RNN similar to other applications of deep learning.

Next, a reduced order modeling technique for subsurface multi-phase flow problems is developed building on the DR-RNN architecture. More specifically, DR-RNN is combined with POD and discrete empirical interpolation method (DEIM) to reduce the computational complexity associated with high-fidelity subsurface multi-phase flow simulations. In the presented formulation, POD is used to construct an optimal set of reduced basis functions and DEIM is employed to evaluate the nonlinear terms independent of the high-fidelity model size. The proposed ROM is demonstrated on two uncertainty quantification test cases involving Monte Carlo simulation of subsurface flow with random permeability field. The obtained results demonstrate that DR-RNN combined with POD-DEIM provides an accurate and stable ROM with a fixed computational budget that is much less than the computational cost of standard POD-Galerkin ROM combined with DEIM for nonlinear dynamical systems.

Finally, this thesis focus on developing multi-fidelity framework to estimate the statistics of high-fidelity model outputs of interest. Recently, Multi-Fidelity Monte Carlo (MFMC) method and Multi-Level Monte Carlo (MLMC) method have shown to significantly accelerate the Monte Carlo estimation by making use of low cost low-fidelity models. In this thesis, the features of both the MFMC method and the MLMC method are combined into a single framework called Multi-Fidelity-Multi-Level Monte Carlo (MFML-MC) method. In MFML-MC method, MLMC framework is developed first in which a multi-level hierarchy of POD approximations of high-fidelity outputs are utilized as low-fidelity models. Next, MFMC method is incorporated into the developed MLMC framework in which the MLMC estimator is modified at each level to benefit from a level specific low-fidelity model. Finally, a variant of deep residual recurrent neural network called Model-Free DR-RNN (MF-DR-RNN) is used as a level specific low-fidelity model in the MFML-MC framework. The performance of MFML-MC method is compared to Monte Carlo

estimation that uses either a high-fidelity model or a single low-fidelity model on two subsurface flow problems with random permeability field. Numerical results show that MFML-MC method provides an unbiased estimator and show speedups by orders of magnitude compared to Monte Carlo estimation that uses a single high-fidelity model.

To Women who work hard physically from day to night throughout the year for less than a pound per day in order to feed their family.

Acknowledgements

First of all, I would like to thank Almighty Allah for providing me complete guidance during my life and providing me the opportunity to write this acknowledgement. Next, I would like to express my sincere gratitude to my supervisors Dr. Ahmed H. Elsheikh and Dr. Florian Doster. Dr. Ahmed H. Elsheikh provided me constant support throughout my stay in Edinburgh. Most of our discussions on my study resembled Generative Adversarial Neural Networks by which I am now confident to generate new research ideas but still rely on Almighty Allah. I would like to thank immensely Ali Danish scholarship and Heriot-Watt University for funding my PhD. I am indebted to madam Debbie Ross for her support. Without her support, I would not have started my journey from India and completed my last six months of my PhD study. I would like to thank all the administrative staff members of Heriot-Watt University especially members of Institute of Petroleum Engineering Department for their support. I would like to thank Sam Paxton for his sincere support.

I would like to acknowledge all my teachers who have dedicated their time for my education. I would like to express my sincere gratitude to brother Dr. Faizal from Saudi Aramco for teaching me how to be humble and polite to everyone inspite of their economic and social status. I would like to thank brother Dr. Eltazy from Sudan for being a nice brotherhood. I am grateful to Heriot-Watt University Muslim Society for providing me the environment that helped me a lot in protecting my Imaan and my Islamic activities. I would like to thank the people of Scotland for making my stay during my PhD study very comfortable. I would like to thank brother Mohamed Alhamadi from Abu Dhabi for his moral support. I would like to thank Shing for his great friendship during my stay in Edinburgh and during the conferences. I am grateful to the student accommodation officer Lucie for providing me a comfortable room in Leonard Horner Hall. I would like to thank Innovative Eye wear from Germany for providing me a wonderful blue light protection computer glass. I would like to thank all the members of Energy Academy building for their friendly behaviour.

I would like to thank my parents for their love and affection. No one had dedicated in the pursuit of my PhD study than my wife, my wife's grandmother Mrs. Fathima and my Daughters. I would like to thank them all for their dedication and moral support.

ACADEMIC REGISTRY

Research Thesis Submission

Name*:*			
School:	EGIS – INSTITUTE OF PETROLEUM ENGINEERING		
Version: *(i.e. First, Resubmission, Final)*		Degree Sought:	PhD

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1) the thesis embodies the results of my own work and has been composed by myself
2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.
6) I confirm that the thesis has been verified against plagiarism via an approved plagiarism detection application e.g. Turnitin.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate*:*		Date:	

Submission

Submitted By *(name in capitals):*	
Signature of Individual Submitting:	
Date Submitted:	

For Completion in the Student Service Centre (SSC)

Received in the SSC by *(name in capitals):*			
Method of Submission *(Handed in to SSC; posted through internal/external mail):*			
*E-thesis Submitted (**mandatory for final theses)***			
Signature:		Date:	

Please note this form should be bound into the submitted thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter will first highlight the necessity for uncertainty quantification, and key challenges in using computational models of subsurface flow for statistical estimation. Following that, brief description will be given on probabilistic methods, and low-fidelity models that can be utilized to propagate uncertainties from the inputs of the flow model to its outputs of interest. Based on the description and the challenges, this chapter will define the scope of this thesis and finally this chapter will present the thesis outline.

1.1 Necessity for Uncertainty Quantification (UQ)

Mathematical models of multi-phase flow through subsurface materials is an essential tool to describe complex physical processes. Generally, the subsurface flows are governed by law of conservations (mass, momentum, and energy) and their mathematical models are represented as a system of nonlinear partial differential equations (PDEs) with suitable initial and boundary conditions [34, 1, 14]. Numerical results from the simulations of subsurface flow models play an important role in a number of engineering applications including ground water management, contaminant transport, and effective extraction of hydrocarbon resources [115, 49]. For example, plume management is a subsurface flow control problem in which the movement of subsurface pollutants is manipulated through the injection or pumping of water. In this plume management task, numerical simulation of subsurface flow and contam-

inant transport is utilized to forecast the spatial distribution of the contaminants at different time instances by which the location and the rate of pumping water are optimized [86, 92]. Another example is the reservoir management for better oil recovery during which it is critical to determine recovery design factors, such as the positions of wells and well injection schedules. To improve the oil recovery over the production life of the reservoir, reservoir simulation is effectively utilized during the development phase as well as during the operation phase [80, 68].

One of the major challenges of subsurface flow model based system management is the effective characterization of subsurface material properties as model parameters (e.g. permeability and porosity). Generally, subsurface properties like permeability of subsurface flow systems are highly heterogeneous [57]. This heterogeneous and multi-scale permeability of natural subsurface flow systems demands comprehensive exploratory drilling, seismic surveys and measurements for the detailed permeability characterization. However, such detailed procedures are not practical as exploratory drilling is expensive and hard to quantify the subsurface properties with current measurement techniques. Hence, heterogeneity and lack of data have lead to uncertainty in the representation of subsurface material properties [57, 77, 92]. Recently, the information obtained from time-lapse seismic data collected from the entire desired subsurface domain are utilized to estimate the parameters of the subsurface flow models [147]. From time-lapse seismic data, it is possible to obtain the dynamic characteristics of the subsurface field like fluid saturation in addition to static properties like pore volume and net gross ratio [117].

The uncertainty in the spatial permeability field can cause unreliable results from the subsurface system models. Therefore, probabilistic methods are needed to effectively propagate the uncertainty in the input data through the high-fidelity mathematical model representing the multi-phase porous media flow. In the context of mathematical modeling, fidelity refers to the degree of representing the physical phenomena of interest. High-fidelity model is a computationally expensive model which describes the physical system of interest with the detailed information and high accuracy. In this thesis, high-fidelity model is refered to a high-dimensional dis-

cretized system of differential equations. The low-fidelity model refers to a simplified model with less detailed and less computational cost compared to the high-fidelity model.

1.2 Probabilistic methods for UQ

As mentioned in the previous section, to properly account for the uncertainties for example during oil reservoir management, probabilistic methods are applied. Uncertainties are present due to many possible sources including (i) parameter uncertainty due to the unavailability of the exact values of input parameters to the mathematical model, (ii) inadequate description of the mathematical model w.r.t the true physics (iii) numerical errors and approximations in the implementation of the simulation code, (iv) inherent variability of some of the inputs to the mathematical model, (v) unavoidable variability of experimental measurements, and (vi) lack of outputs for all possible inputs due to limited resources [105]. In this thesis work, we focus on parametric uncertainties (e.g., permeability, porosity values, rock properties) and represent those uncertain parameters as random variables with appropriate probability distribution functions (e.g., lognormal distribution function). We are mainly concerned with the efficient computation of the statistics of the high-fidelity model outputs (quantities of interest (QoI)).

Monte Carlo method (MC) is a popular and highly successful probabilistic method to approximate the statistics of the desired QoI. In Monte Carlo method, first, N (e.g., 10^5) independent realizations of the random input are sampled. Next, the deterministic high-fidelity model is solved for each random input sample and thus N number of high-fidelity solutions are obtained. Finally, the desired statistical properties of the solution are estimated from the set of high-fidelity data of size N. Unfortunately, the accuracy of the estimators performed via the MC method depends on the inverse square root of the number of simulations performed [63, 64]. Hence, MC method requires a substantial number of samples to converge the estimations towards the desired statistics. In addition, if a single realization of high-fidelity simulation is computationally expensive, then MC method becomes computationally

prohibitive for large-scale realistic problems.

Polynomial Chaos expansion (PCE) based method is an alternative approach to MC method that aims to approximate the high-fidelity model output (say $\mathbf{y} \in \mathbb{R}^n$) of interest using PCE with respect to the uncertain input parameters [60]. The type of polynomial basis function used in PCE depends on the type of probability distribution function (PDF) associated with the input random field. For example, Hermite polynomial is associated with the Gaussian input random variable, Laguerre polynomials with a Gamma distributed variable, Jacobi polynomials with Beta distribution, etc [144]. Although it is required to expand the high-fidelity model output as infinite series in PCE method, for practical simulations, the PCE series has to be truncated to a finite number of terms say N_{pc} determined from $N_{pc} + 1 = \frac{(n_{pc}+p_{pc})!}{n_{pc}!p_{pc}!}$, where p_{pc} is the order of the expansion and n_{pc} is the dimensionality of the random input vector [93, 10]. PCE probabilistic method is mainly classified based on the way the coefficients of expansion in the PCE truncated series are determined: Intrusive method and Non intrusive method. Intrusive methods involve suitable spatial discretization scheme of the high-fidelity governing equations and substitution of the PCE approximations into the discretized equations to obtain an algebraic system of size $n \cdot N_{pc} \times n \cdot N_{pc}$ [60, 127]. After solving this system for the coefficients in the expansion, any component (y_i) of the model output ($\mathbf{y}$) is characterized as a random variable with N_{pc} number of coefficients [127]. Hence, the amount of computation required for solving the resultant system for the coefficients is thus much greater than that required for the deterministic analysis of the same high-fidelity model output. In non intrusive PCE approaches, the coefficients can be computed by two approaches (i) spectral projection or collocation approach and (ii) least-square regression approach. In spectral projection method, coefficients can be directly obtained by projecting the PCE of the model output onto each basis function [10, 91]. By employing the orthonormal relation between the PCE basis functions, coefficients can be computed as the expectation of the product of the model output and the corresponding basis function. A popular approach to approximate the aforementioned expectation/integral is to utilize a suitable cubature rule

(integration rule) with certain number of nodal points (model output) and corresponding weights. The most direct way to construct cubature rules is to take (full) tensor products of one dimensional quadrature rules. However, as the expectation involved to compute coefficients typically corresponds to a high dimensional integration problem, the straight-forward tensorization normally results in cubatures which heavily suffer from the curse of dimensionality, i.e., the number of nodes is infeasibly large for practical computations. The curse of dimensionality can be alleviated by applying sparse-grid cubatures, often obtained by employing sparse tensor or dimension-adaptive sparse-grid constructions [93]. The second approach to calculate the coefficients is through solving a least-squares problem that is constructed after evaluating the model on a set of samples N_{ls} from the uncertainty space. Generally, the number of samples in the least-squares problem should be greater than the number of unknown PCE coefficients (a value $N_{lc} \approx 2\ (N_{pc} + 1)$ is considered to be sufficient) for a robust solution and to prevent overfitting [93, 91]. Since computing a single high-fidelity model output is computationally expensive, sufficiently high N_{pc} can greatly reduce the efficiency of such approaches.

Another viable approach for standard MC method is to use computationally inexpensive low-fidelity models. Traditionally, low-fidelity models are utilized in the MC framework in order to completely replace the high-fidelity model. This low-fidelity approach approximately estimates the statistics of the QoI with orders of magnitude less CPU time. However, it requires stringent condition on the low-fidelity models such as the absolute errors of the low-fidelity model outputs with respect to the high-fidelity model outputs should be very low and the low-fidelity models should be computationally very cheap [113]. Recently, instead of utilizing low-fidelity models to replace the high-fidelity model, the low-fidelity models are utilized to accelerate the convergence of the standard MC method [64, 105, 113]. More precisely, in this multi-fidelity approach, a significant portion of the computation is relegated to the inexpensive low-fidelity models with only occasional recourse to the expensive high-fidelity model as a correction. This approach does not pose stringent condition on the low-fidelity model as mentioned earlier. However, the key require-

ment for most multi-fidelity methods to be effective is that the low-fidelity model outputs should have some sort of similar trends to the high-fidelity model outputs like the degree of correlation between the low-fidelity model and the high-fidelity model [64, 105, 113]. In addition, a perfectly correlated low-fidelity model must also be cheaper to evaluate than the high-fidelity model to accelerate the convergence of the standard MC method that uses the high-fidelity model only. Recently, Multi-Level Monte Carlo (MLMC) and Multi-Fidelity Monte Carlo (MFMC) methods has provided a multi-fidelity framework to exploit low-fidelity models for the acceleration of the standard MC method that uses the high-fidelity model only. In MLMC method, we generally use a multi-level hierarchy of spatial and/or temporal discretizations to construct low-fidelity models. In many multi-phase flow problems, we face difficulties to obtain fast and consistent upscaling of permeability fields onto coarser grids and also we face difficulties to accurately specify complicated boundary conditions of the high-fidelity model on coarse grids. Hence, in such situations, it appears very difficult to efficiently apply standard MLMC approaches for UQ. In MFMC method, we can exploit low-fidelity models of any type for the acceleration of the Monte Carlo estimation that uses high-fidelity model only. However, it should be noted that, MFMC methods are mainly tailored for scalar QoI.

1.3 Low-fidelity models for UQ

The primary factor behind the synthesis of low-fidelity models is to consider the optimal trade-off between the computational cost and the accuracy of the high-fidelity model. More precisely, low-fidelity modelling techniques are aimed to design computationally low budget models that can effectively approximate the input-output relations of the high-fidelity model. The methods to obtain low-fidelty models can be broadly classified into three categories namely, (i) simplified physics or upscaling methods [43], (ii) data driven black-box methods [53, 145], and (iii) projection based reduced order models commonly referred to as ROM [17, 89, 5].

Upscaling technique is designed to generate a coarse grid description which is approximately equivalent to the underlying fine grid description of high-fidelity

model [43]. Within the context of subsurface flow simulation, upscaling techniques are utilized to capture the average large scale flow features and the effect of the small scale features in the heterogeneous permeability field is then approximately modeled. Durlofsky [42] has provided comprehensive reviews that describe a variety of upscaling methods. Although, upscaling is a popular technique to generate low-fidelity models, it is still not effectively developed particularly for multi-phase flow problems in highly heterogeneous subsurface properties [43, 33]. For example, most of the existing efforts in upscaling technique restrict to the assumption that the small scales have a periodic structure [139, 33]. With this assumption, one face difficulties to derive governing equations for the large scale average and the small scale fluctuations in subsurface flow problems with many different length scales. To be noted, developing a computationally effective downscaling procedure in order to downscale (un-upscale) the results obtained from the upscaling technique is also equally challenging.

Data driven or black-box methods builds low-fidelity models directly from the high-fidelity simulation results and therefore does not require the knowledge of the governing equations or the underlying physics of the model [53, 145]. In data driven methods, the training samples of high-fidelity data are utilized to build a regression model to interpolate the input-output relationship. For example, in the context of subsurface flow simulation, the inputs parameters can be the parameters describing the uncertain permeability field, while the output can be the time series of oil production rate. Although the trained regression model is computationally more efficient than to run the high-fidelity model, their performance depend on how effectively the training samples cover the input parameter space. In addition, as the dimensionality of the input space increases, the computational complexity of training process increases (i.e. time to build the data-driven low-fidelity model) and greatly undermines the applicability of black-box technique in many real world problems, where a lot of input parameters are involved [53, 145].

In projection based model reduction method, a reduced order model of size $r \ll n$ is built which shares the same mathematical structure of the high-fidelity model

(full order model (FOM)) [89, 5]. The fundamental criteria for such reduction is the assumption that the large scale system's dynamics are evolved in a low dimensional manifold of a high dimensional state space [53, 89, 5, 149]. All projection based model reduction techniques essentially involve two ingredients. The first ingredient is the construction of the basis functions that spans the low dimensional subspace. The second ingredient is projecting the governing equations of the system onto a low dimensional subspace. The aforementioned two ingredients are accomplished by computing a transformation matrix $\mathbf{U} \in \mathbb{R}^{n \times r}$, and a projection matrix $\mathbf{W} \in \mathbb{R}^{n \times r}$. The model reduction technique with $\mathbf{W} = \mathbf{U}$ is called Galerkin projection technique and with $\mathbf{W} \neq \mathbf{U}$ and $\mathbf{W}^\top \mathbf{U} = \mathbf{I}_r \in \mathbb{R}^{r \times r}$ is called a Petrov-Galerkin projection technique. Here, $\mathbf{I}_r$ is the identity matrix.

Projection based model reductions techniques could also be classified based on the way $\mathbf{U}$ and $\mathbf{W}$ matrices are computed. The system based methods such as balanced truncation [66], Krylov-subspace methods [54] are mainly developed for linear time invariant problems and construct the basis matrices from their system matrices. Snapshot based methods such as reduced-basis methods [121], and proper orthogonal decomposition [125, 17] construct the basis matrices ($\mathbf{U}$ and $\mathbf{W}$) from a set of high-fidelity simulation results (training snapshots) obtained by solving the full order model at selected points in the input parameter space.

Among the aforementioned model reduction techniques, POD coupled with Galerkin projection known as POD-Galerkin ROM has been widely applied to many large-scale nonlinear systems. However, the effectiveness of POD-Galerkin ROM in handling nonlinear systems is limited mainly by two factors. The first factor is related to the treatment of the nonlinear terms in the POD-Galerkin ROM [30, 119, 27] and the second factor is related to maintaining the overall stability of the resulting ROM [27, 67, 68, 24, 137]. In relation to computing reduced non-polynomial nonlinear functions, POD based ROMs is usually dependent on the FOM state variables and henceforth, the computational cost of evaluating the ROM is still a function of FOM dimension. Several techniques have been developed to reduce the computational cost of evaluating the nonlinear term in POD ROMs including trajectory

piecewise linearization (TPWL) [119], gappy POD technique [140], Missing Point Estimation (MPE) [12], Best Point Interpolation Method [106], and Discrete Empirical Interpolation Method (DEIM) [12, 30]. Among these techniques, TPWL and DEIM are widely used for efficient treatment of nonlinearities in multi-phase flow reservoir simulations [61, 67, 68]. The overall convergence and stability is another issue that limits the applicability of POD-Galerkin based ROMs. POD-Galerkin projection methods manage to decrease the computational complexity by orders of magnitude as a result of state variable's dimension reduction. However, this reduction goes hand in hand with a loss in accuracy. Moreover, slow convergence and in some cases model instabilities [137, 67, 24] are observed as the errors in the reduced state variables are propagated in time. More specifically, the performance of POD-Galerkin ROMs is directly influenced by the number of POD basis used in the POD-Galerkin projection. However, in many applications involving nonlinear conservation laws (e.g. high Reynold number fluid flow), POD-Galerkin ROM has shown poor performance even after retaining a sufficient number of POD basis [137, 125, 17, 40]. Several stabilization techniques have been proposed in the recent literature to enhance the stability of POD-Galerkin ROM including (i) closing the POD ROM using a set of closure models similar to those adopted in standard turbulence modeling [17, 137], (ii) to compute a new set of optimal basis functions or to improve the POD basis vectors by solving a constrained matrix optimization problem [67, 68], (iii) to address the stability and nonlinearity issues of POD based ROMs by developing non-intrusive POD ROMs [143, 136], where the data-fit models are used to regress the relationship between the input parameter and the reduced representation of the FOM state vector. However, all the aforementioned POD stabilization techniques [137, 67, 68] are not cost effective and ultimately do not guarantee stability of the extracted reduced order models.

1.4 Thesis Scope

Based on the description from the previous sections, there is a necessity to find effective low-fidelity models, and an effective multi-fidelity approach that optimally

exploits a hierarchy of low-fidelity models on UQ tasks. In addition, the use of these representative low-fidelity models within multi-fidelity uncertainty quantification framework should be applied to a wide range of nonlinear dynamical systems, especially those representing large scale subsurface flows. Therefore, the objective of this thesis is to derive computationally efficient low-fidelity models and to extend the existing multi-fidelity MC frameworks in order to overcome some of the computational and accuracy limitations encountered in the traditional UQ methods. In particular, this thesis will

- Develop a physics aware recurrent neural network (RNN) architecture as an effective low-fidelity model.
- Compare the efficiency and the performance of the developed physics aware RNN with the standard RNN [107, 79, 94] on UQ problems involving nonlinear dynamical systems.
- Develop a strategy to construct a robust reduced order model combining the developed RNN with POD-Galerkin methods.
- Demonstrate the validity of the developed reduced order model as an effective low-fidelity model over a range of input parameter variations in the high-fidelity model.
- Demonstrate the potential benefit of the developed reduced order model for UQ problems involving multi-phase porous media flow.
- Compare the efficiency and the performance of the developed reduced order model with the standard POD-Galerkin reduced order models for UQ.
- Develop a multi-fidelity MC method called MFML-MC method that combine the features of both the Multi-Fidelity Monte Carlo method and Multi-Level Monte-Carlo method.
- Develop a systematic strategy to embed a set of hierarchical POD models into the developed MFML-MC method.

- Demonstrate the potential benefit of the MFML-MC method for the vector valued time series QoI.

1.5 Thesis Outline

The contribution and the outline of this thesis is listed as follows. In Chapter 2, we develop a physics aware RNN architecture called Deep Residual Recurrent Neural Network [104] as an effective low-fidelity model. The architecture of DR-RNN is constructed to account for the dynamics of the high-fidelity model. We first use DR-RNN for reducing the computational complexity of nonlinear ODE systems from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$. More precisely, computational cost of an algorithm is measured in terms of floating point operations (flops). For example, one flop is counted as one addition, subtraction, multiplication, division, squareroot operation. When counting flops to estimate computational cost, it is standard practice to focus on the part of the algorithm which dominates the computational work. In solving nonlinear ODE systems for example by Newton method (see chapter 2.3 for more details), the computational work is dominated by dense matrix equation solver which cost approximately n^3 flops. In DR-RNN, the dominative computational work is matrix-vector multiplication and it take nearly n^2 flops to do one dense matrix-vector multiplication. In this context, we evaluate DR-RNN in emulating nonlinear dynamical systems using a different number of large time step sizes violating the numerical stability condition for a small time discretization errors. We show how the accuracy of DR-RNN is increased as the number of neural network layer increases. Next, we use DR-RNN for spatial dimensionality reduction of dynamical systems governed by time dependent PDEs with parametric uncertainty. In this context, we use DR-RNN to approximate ROM derived from a POD-Galerkin strategy. For the nonlinear case, we combine POD with the DEIM algorithm for approximating the nonlinear function. We show that DR-RNN can provide a significant reduction of the computational complexity of the extracted ROM limiting the computational complexity to $\mathcal{O}(K \times r^2)$ instead of $\mathcal{O}(p \times r^3)$ per time step for the nonlinear POD-DEIM method, where $K \ll p$ is the number of stacked network layers in DR-RNN

and p is the number of Newton iterations in POD-DEIM.

In Chapter 3, we extend the DR-RNN into nonlinear multi-phase flow problems with distributed uncertain parameters. In this extended formulation, we construct a DR-RNN architecture termed DR-RNN$^{\text{pd}}$ which accounts the dynamics of POD-DEIM ROM. We evaluate the proposed DR-RNN$^{\text{pd}}$ on two forward uncertainty quantification problems involving two phase flow in subsurface porous media. The uncertainty parameter is the permeability field modeled as log-normal distribution. In the two test cases, we show that DR-RNN$^{\text{pd}}$ can provide accurate and stable approximations of the high-fidelity model in comparison to the standard POD ROM.

In Chapter 4, we combine the features of both the Multi-Fidelity Monte Carlo method and Multi-Level Monte Carlo method into a single framework called MFML-MC method. In MFML-MC method, we first leverage a sequence of POD based approximations of the QoI as low-fidelity models. We next incorporate the developed low-fidelity models into Multi-Level Monte Carlo framework. We then incorporate Multi-Fidelity Monte Carlo method into Multi-Level Monte Carlo framework in which the Multi-Level Monte Carlo estimator is modified at each level to benefit from a level specific low-fidelity model. Finaly, We use a variant of DR-RNN called MF-DR-RNN as a level specific low-fidelity model in the MFML-MC framework. We compare the performance of MFML-MC method to Monte Carlo estimation that uses either a high-fidelity model or a single low-fidelity model only. Finally, Chapter 5 summarizes all the work, and concludes with some suggestions for future work.

Chapter 2

DR-RNN: A deep residual recurrent neural network for model reduction [1]

2.1 Introduction

Recently, detailed numerical simulations of highly nonlinear partial differential equations representing multi-physics problems became possible due to the increased power and memory of modern computers. Nevertheless, detailed simulations remain far too expensive to be used in various engineering tasks including design optimization, uncertainty quantification, and real-time decision support. For example, Bayesian calibration of subsurface reservoirs might involve millions of numerical simulations to account for the heterogeneities in the permeability fields [48, 49]. Model Order Reduction (MOR) provides a solution to this problem by learning a computationally cheap model from a set of the detailed simulation runs. These reduced models are used to replace the high-fidelity models in optimization and statistical inference tasks. MOR could be broadly categorized into three different classes: simplified physics based models, data-fit black box models (surrogate models) [115] and projection based reduced order models commonly referred to as ROM [53].

[1] The contents of this chapter has been published in Nagoor Kani and Elsheikh [104] and has been submitted to Mathematical Geosciences journal.

Physics based reduced order models are derived from high-fidelity models using approaches such as simplifying physics assumptions, using coarse grids, and/or upscaling of the model parameters. Data-fit models are generated using regression of the high-fidelity simulation data from the input to the output [53, 115]. In projection based ROM, the governing equations of the system are projected into a low-dimensional subspace spanned by a small number of basis functions commonly obtained by Galerkin projection. In all projection based ROM methods, it is generally assumed that the main solution characteristics could be efficiently represented using a linear combination of only a small number of basis functions. Under this assumption, it is possible to accurately capture the input-output relationship of a large-scale full-order model (FOM) using a reduced system with significantly fewer degrees of freedom [89, 17].

In projection based ROM, different methods could be used to construct the projection bases including: Proper Orthogonal Decomposition (POD), Krylov subspace methods, and methods based on truncated balanced realization [89, 119]. ROM based on Proper Orthogonal Decomposition has been widely used to model nonlinear systems [53, 119]. Despite the success of POD based methods, there exist a number of outstanding issues that limit the applicability of POD method as an effective reduced order modeling technique.

One issue is related to the cost of evaluating the projected nonlinear function and the corresponding Jacobian matrix in every Newton iteration. The reason is in order to compute the reduced approximation of the general nonlinear function of the FOM system, one has to reconstruct the FOM state variable from its reduced approximation which causes computational cost on the order of FOM dimension. These costs create a computational bottleneck that reduces the performance of the resulting reduced order models. Some existing approaches for constructing a reduced order approximation to alleviate such computational bottleneck are gappy POD technique, sparse sampling, Missing Point Estimation (MPE), Best Point Interpolation Method (BPIM), Empirical Interpolation Method and Discrete Empirical Interpolation Method (DEIM) [140, 12, 30]. All these methods rely on interpolation schemes

involving the selection of discrete spatial points for producing an interpolated approximation of the nonlinear functions. Moreover, these methods are developed especially for removing the computational complexity due to the nonlinear function in the PDE system after spatial discretization.

Another issue is related to convergence and stability of the extracted ROM. Although POD based methods decrease the calculation times by orders of magnitude as a result of reducing the state variables dimension, this reduction goes hand in hand with loss of accuracy. This may result not only in inaccurate results, but also in slow convergence and in some cases model instabilities. Slow convergence means that many iterations are needed to reach the final solution and corresponds to an increase in the computational time. Divergence is even less desirable as it produces invalid simulation results.

Artificial Neural Networks (ANN) have found growing success in many machine learning applications such as computer vision, speech recognition and machine translation [73, 70, 74, 65]. Further, ANNs offer a promising direction for the development of innovative model reduction strategies. Neural network use in the domain of MOR is generally limited to constructing surrogate models to emulate the input-output relationship of the system based on the available simulation and experimental data [87, 126]. Neural networks have also been combined with POD to generate reduced order models without any knowledge of the governing dynamical systems [141]. One reason for developing such non-intrusive reduced order modeling methods is to address the main issues of POD-Galerkin projection ROM technique such as stability and efficient nonlinearity reduction.

Recently, Recurrent Neural Network (RNN) a class of artificial neural network where connections between units form a directed cycle have been successfully applied to various sequence modeling tasks such as automatic speech recognition and system identification of time series data [73, 70, 74, 65]. RNN has been used to emulate the evolution of dynamical systems in a number of applications [150, 11] and hence has large potential in building surrogate models and reduced order models for nonlinear dynamical systems. The standard approach of modeling dynamical systems using

RNN relies on three steps: (a) generating training samples from a number of detailed numerical simulations, (b) defining the suitable structure of RNN to represent the system evolution, and (c) fitting the RNN parameters to the training data. This pure data-driven approach is very general and can be effectively tuned to capture any nonlinear discrete dynamical system. However, the accuracy of this approach relies on the number of training samples (obtained by running a computationally expensive model) and on the selected RNN architecture. In addition, generic architectures might require a large number of parameters to fit the training data and thus increases the computational cost of the RNN calibration process.

Many types of recurrent neural network architectures have been proposed for modeling time-dependent phenomena [150, 11]. Among those, a recurrent neural network called Error Correction Neural Network (ECNN) [150], that utilizes the misfit between the model output and the true output termed as model error to construct the RNN architecture. ECNN architecture [150] augmented the standard RNN architecture by adding a correction factor based on the model error. Further, the correction factor in ECNN was activated only during the training of RNN. In other words, ECNN takes the time series of the reference output as an input to RNN for a certain length of the time period and after that time period (i.e. in future time steps), ECNN forecasts the output without the reference output as input from the fitted model.

In this chapter, we propose a physics aware RNN architecture to capture the underlying mathematical structure of the dynamical system under consideration. We further extend this architecture as a deep residual RNN (DR-RNN) inspired by the iterative line search methods [18, 130] which iteratively find the minimiser of a nonlinear objective function. The developed DR-RNN is trained to find the residual minimiser of numerically discretized ODEs or PDEs. We note that the concept of depth in the proposed DR-RNN is different from the view of hierarchically representing the abstract input to fit the desired output commonly adopted in standard deep neural network architectures [107, 108]. More precisely, one of the fundamental hypothesis of deep learning technique (standard neural network

with many number of layers) is on increasing the number of network layers increases the input-output relation in neural network architecture whereas in (DR-RNN), increasing the number of layers increases the convergence of the (DR-RNN) output towards the desired output. The proposed DR-RNN method reduces the computational complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ for fully coupled nonlinear systems of size n and from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ for sparse nonlinear systems obtained from discretizing time-dependent partial differential equations.

We further combined DR-RNN with projection based ROM ideas (e.g. POD and DEIM [30]) to produce an efficient explicit nonlinear model reduction technique with superior convergence and stability properties. Combining DR-RNN with POD/DEIM, resulted in further reduction of the computational complexity form $\mathcal{O}(r^3)$ to $\mathcal{O}(r^2)$, where r is the size of the reduced order model.

The rest of this chapter is organized as follows: Section 2.1 describes dimension reduction via POD-Galerkin method followed by a discussion of DEIM in section 2.2. In Section 3, we present a brief background overview of deep neural networks (feedforward and recurrent), then we introduce the proposed DR-RNN in section 4. In section 5, we evaluate the proposed DR-RNN on a number of test cases. Finally, in Section 6 the conclusions of this manuscript are presented.

2.2 Background for Model Reduction

In this section, we first define the class of dynamical systems to be considered in this study. Following that, we present a general framework for reduced order modeling based on the concept of projecting the original state space into a low-dimensional, reduced-order space. At this point, we also discuss the computational bottleneck associated with dimensionality reduction for general nonlinear systems. Then we present the DEIM algorithm to reduce the time complexity of evaluating the nonlinear terms.

2.2.1 POD-Galerkin

We consider a general system of nonlinear differential equations of the form:

$$\frac{d\mathbf{y}}{dt} = \mathbf{A}\ \mathbf{y} + \mathbf{F}(\mathbf{y}) \tag{2.1}$$

where $\mathbf{y}(\mathbf{a}, t) \in \mathbb{R}^n$ is the state variable at time t and $\mathbf{a} \in \mathbb{R}^d$ is a system parameter vector. The linear part of the dynamical system is given by the matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and the vector $\mathbf{F}(\mathbf{y}) \in \mathbb{R}^n$ is the nonlinear term. The nonlinear function $\mathbf{F}(\mathbf{y})$ is evaluated component-wise at the n components of the state variable $\mathbf{y}(\mathbf{a}, t)$. The complete space of $\mathbf{y}$ is spanned by a set of n orthonormal basis vectors $\mathcal{U} = \text{span}(\mathbf{u}_1\ \cdots\ \mathbf{u}_n)$. Since $\mathbf{y}$ is assumed to be in a certain low dimensional subspace $\tilde{\mathcal{U}} \subset \mathcal{U}$, all the solutions of Eq. 2.1 could be expressed in terms of only r basis vectors ($r \ll n$) that span $\tilde{\mathcal{U}}$. The solution $\mathbf{y}(\mathbf{a}, t)$ could then be approximated as a linear combination of these basis vectors as:

$$\mathbf{y} = \mathbf{U}_r\ \tilde{\mathbf{y}} + \mathbf{r}_{\text{POD}} \tag{2.2}$$

where $\mathbf{r}_{\text{POD}}$ is the residual representing the part of the $\mathbf{y}$ that is orthogonal to the subspace $\tilde{\mathcal{U}}$. Thus, the inner product of $\mathbf{r}_{\text{POD}}$ with any of the basis vectors that span $\tilde{\mathcal{U}}$ is zero (i.e. $\mathbf{U}_r^\top\ \mathbf{r}_{\text{POD}} = 0$). The basis vectors of $\tilde{\mathcal{U}}$ are collected in the matrix $\mathbf{U}_r \in \mathbb{R}^{n\times r}$ and $\tilde{\mathbf{y}}(t) \in \mathbb{R}^r$ is the time-dependent coefficient vector. POD identifies the subspace $\tilde{\mathcal{U}}$ from the singular value decomposition (SVD) of a series of temporal snapshots of the full order system (Eq. 2.1) collected in the snapshot matrix $\mathbf{X} = [(\mathbf{y}_1\ \cdots\ \mathbf{y}_T)^1\ \cdots\ (\mathbf{y}_1\ \cdots\ \mathbf{y}_T)^L] \in \mathbb{R}^{n\times(T\cdot L)}$, where L is the number of different simulation runs (i.e. different initial conditions, different controls and/or different model parameters). The SVD of $\mathbf{X}$ is computed as:

$$\mathbf{X} = \mathbf{U}\ \Sigma\ \mathbf{W}^* \tag{2.3}$$

$\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3\ \cdots\ \mathbf{u}_n] \in \mathbb{R}^{n\times n}$ is the left singular matrix and $\Sigma = \text{diag}(\sigma_1 > \sigma_2 > \sigma_3 > \cdots\ \sigma_{ns} \geq 0)$ is the diagonal matrix containing the singular values σ_i of the snapshot matrix $\mathbf{X}$ in descending order. SVD is a matrix factorization

with a geometrical property that under the transformation of SVD, a sphere is mapped to an ellipse in a n dimensional space. More precisely, SVD stretches a unit sphere in $\mathbb{R}^n$ to a hyperellipse in $\mathbb{R}^n$ by factors equivalent to the singular values in the orthogonal directions represented by $\{\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3\ \cdots\ \mathbf{u}_n\}$. Thus the vectors $\{\sigma_1\mathbf{u}_1\ \sigma_2\mathbf{u}_2\ \sigma_3\mathbf{u}_3\ \cdots\ \sigma_n\mathbf{u}_n\}$ represents the principal semiaxes of the hyperellipse. Figure 2.1 displays the pictorial representation of the transformation under SVD where an unit sphere is rotated, and stretched to a hyperellipse.

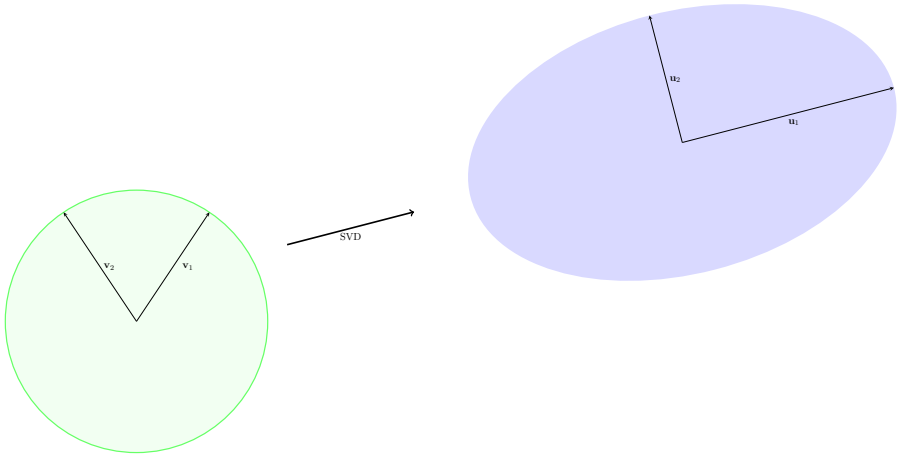


Figure 2.1: Pictorial representation of a SVD transformation where a sphere is mapped to an ellipse in a two dimensional space.

The orthonormal basis matrix $\mathbf{U}_r$ for approximating $\mathbf{y}(\mathbf{a}, t)$ is given by the first r columns of the matrix $\mathbf{U}$. Substituting Eq. 2.2 into Eq. 2.1 while neglecting $\mathbf{r}_{\text{POD}}$, one gets:

$$\frac{d(\mathbf{U}_r\ \tilde{\mathbf{y}})}{dt} = \mathbf{A}\ \mathbf{U}_r\ \tilde{\mathbf{y}} + \mathbf{F}(\mathbf{U}_r\ \tilde{\mathbf{y}}). \tag{2.4}$$

By multiplying Eq. 2.4 with $\mathbf{U}_r^\top$, one obtains POD based ROM defined by:

$$\frac{d\tilde{\mathbf{y}}}{dt} = \tilde{\mathbf{A}}\ \tilde{\mathbf{y}} + \mathbf{U}_r^\top\ \mathbf{F}(\mathbf{U}_r\ \tilde{\mathbf{y}}) \tag{2.5}$$

where $\tilde{\mathbf{A}} = \mathbf{U}_r^\top\ \mathbf{A}\ \mathbf{U}_r$. We note that the POD-Galerkin ROM (Eq. 2.5) is of reduced dimension $r \ll n$ and could be used to approximate the solution of the high-dimensional full order model (Eq. 2.1). The computation of the first term in the right hand side of Eq. 2.5 involves r^2 operations in comparison to n^2 multiplications in the FOM. However, the nonlinear term $\mathbf{U}_r^\top\ \mathbf{F}(\mathbf{U}_r\ \tilde{\mathbf{y}})$ cannot be simplified to an $\mathcal{O}(r)$

nonlinear evaluations. In the following subsection, we review the Discrete Empirical Interpolation Method (DEIM) which is aimed at approximating the nonlinear term $\mathbf{F}(\mathbf{y})$ in Eq. 2.5 using $m \ll n$ evaluations, where m is the dimension of the nonlinear term in a reduced order system and thus rendering the solution procedure of the reduced order system independent of the high-dimensional system size n.

2.2.2 DEIM

As outlined in the previous section, evaluating the nonlinear term $\mathbf{F}(\mathbf{U}_r\ \tilde{\mathbf{y}})$ in the POD-Galerkin method is still an expensive computational step, as the inner products of the full high-dimensional system is needed. The DEIM algorithm tries to reduce the complexity of evaluating the nonlinear terms in the POD based ROM (Eq. 2.5) by computing the nonlinear term only at m carefully selected locations and interpolate everywhere else. The nonlinear term $\mathbf{F}$ in Eq. 2.1 is approximated by a subspace spanned by an additional set of orthonormal basis represented as $\tilde{\mathcal{V}} = [\mathbf{v}_1\ \cdots\ \mathbf{v}_n]$. More specifically, a low-rank representation of the nonlinearity is computed using singular value decomposition of a snapshot matrix of the nonlinear function resulting in:

$$\mathbf{X_F} = \mathbf{V}\ \Sigma_{\mathbf{F}}\ \mathbf{W}^*_{\mathbf{F}} \tag{2.6}$$

where, $\mathbf{X_F}$ is the snapshot matrix of the nonlinear function evaluated using the sample solutions $\mathbf{y}(a,t)$ directly from the snapshot solution matrix $\mathbf{X}$ defined in the previous section, $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\ \cdots\ \mathbf{v}_n] \in \mathbb{R}^{n\times n}$ is the left singular matrix and $\Sigma_{\mathbf{F}}$ is the diagonal matrix containing the singular values of the snapshot matrix $\mathbf{X_F}$ in descending order, and $\mathbf{W_F}$ is the right singular matrix of $\mathbf{X_F}$. The m-dimensional basis for optimally approximating $\mathbf{F}(\mathbf{y})$ given by the first m columns of the matrix $\mathbf{V}$, denoted by $\mathbf{V}_m$. The nonlinearity vector $\mathbf{F}$ is then approximated as:

$$\mathbf{F} \approx \mathbf{V}_m\ \tilde{\mathbf{f}} \tag{2.7}$$

where $\tilde{\mathbf{f}}(\mathbf{a},t)$ is similar to $\tilde{\mathbf{y}}(\mathbf{a},t)$ in Eq. 2.2. The idea behind DEIM is to make an optimal selection of m rows in $\mathbf{V}_m$ such that the original over-determined system

Eq. 2.7 is approximated by an invertible system with an error as small as possible. The selection procedure described in [30] is performed to determine the boolean matrix $\mathbf{P} = [\mathbf{e}_{\phi_1} \ \cdots \ \mathbf{e}_{\phi_m}] \in \mathbb{R}^{n \times m}$ while making use of the orthonormal basis vectors $\mathbf{V}_m = [\mathbf{v}_1 \ \cdots \ \mathbf{v}_m]$. The columns of the boolean matrix $\mathbf{P}$ are specific columns of n dimensional identity matrix [30]. Using $\mathbf{P}$, the basis interpolation of Eq. 2.7 can be made invertible and thus solvable for $\tilde{\mathbf{f}}(\mathbf{a}, t)$

$$\mathbf{P}^\top \mathbf{F} \approx (\mathbf{P}^\top \mathbf{V}_m)\tilde{\mathbf{f}} \qquad \Leftrightarrow \qquad \tilde{\mathbf{f}} \approx (\mathbf{P}^\top \mathbf{V}_m)^{-1}\mathbf{P}^\top \mathbf{F} \tag{2.8}$$

Using this expression of $\tilde{\mathbf{f}}(\mathbf{a}, t)$, the approximate nonlinear term $\mathbf{F}(\mathbf{U}_r \ \tilde{\mathbf{y}})$ in Eq. 2.7 is formulated as:

$$\mathbf{F} \approx \mathbf{V}_m \cdot (\mathbf{P}^\top \mathbf{V}_m)^{-1}\mathbf{P}^\top \cdot \mathbf{F}(a, \mathbf{y}) \approx \mathbf{V}_m(\mathbf{P}^\top \mathbf{V}_m)^{-1} \cdot \mathbf{F}(\mathbf{P}^\top \mathbf{U}_r \tilde{\mathbf{y}}) = \mathbf{D} \cdot \mathbf{F}(\mathbf{P}^\top \mathbf{U}_r \tilde{\mathbf{y}}) \tag{2.9}$$

where $\mathbf{D} = \mathbf{V}_m(\mathbf{P}^\top \mathbf{V}_m)^{-1}$ is referred to as the DEIM-matrix. Due to the selection by $\mathbf{P}$, only m components of the right-side $\mathbf{F}$ are needed. In addition, for nonlinear dynamical systems, implicit time integration schemes are often used. This leads to a system of nonlinear equations that must be solved at each time step for example using Newton's method. At each iteration, besides the nonlinear term $\mathbf{F}$, the Jacobian $\mathbf{J_F}$ of the nonlinear term must also be computed with a computational cost depending on the full order dimension n during the evaluation of the reduced Jacobian matrix $\tilde{\mathbf{J}}_\mathbf{F}$ defined by,

$$\tilde{\mathbf{J}}_\mathbf{F} = \mathbf{U}_r^\top \ \mathbf{J_F}(\mathbf{y}) \ \mathbf{U}_r \tag{2.10}$$

Similar to the approximation of $\mathbf{F}$ by DEIM method, the approximation for the reduced Jacobian $\tilde{\mathbf{J}}_\mathbf{F}$ of the nonlinear term using DEIM takes the form [30]:

$$\tilde{\mathbf{J}}_\mathbf{F} \approx \mathbf{U}_r^\top \mathbf{V}_m(\mathbf{P}^\top \mathbf{V}_m)^{-1}\mathbf{J_F}(\mathbf{P}^\top \mathbf{U}_r \tilde{\mathbf{y}})\mathbf{P}^\top \mathbf{U}_r \tag{2.11}$$

In summary, by augmenting the standard POD formulation with DEIM, we can derive the POD-DEIM reduced order model of the form:

$$\frac{d\tilde{\mathbf{y}}}{dt} = \tilde{\mathbf{A}}\ \tilde{\mathbf{y}} + \mathbf{D} \cdot \mathbf{F}(\mathbf{P}^\top \mathbf{U}_r \tilde{\mathbf{y}}) \tag{2.12}$$

2.3 Review of standard RNN architectures

In this section, we briefly present the basic architecture of deep neural networks. Following that, we review standard architectures of recurrent neural networks and discuss its ability to approximate any dynamical system supported by universal approximation theorem. Then, we discuss the difficulties of training RNNs due to the vanishing gradient problem. Finally, we introduce the Long Short Term Memory (LSTM) architecture as a standard method to overcome the vanishing gradient problem in RNNs.

2.3.1 Deep Feedforward Neural Network

Artificial Neural Network (ANN) is a machine learning method that expresses the input-output relationship of the form:

$$\mathbf{y} = \mathbf{y}^{(\text{ANN})} = \mathbf{W}^{\text{ANN}^\top}\ \phi_h(\mathbf{U}^{\text{ANN}^\top}\ \bar{\mathbf{x}}) + \boldsymbol{\eta} \tag{2.13}$$

where $\bar{\mathbf{x}} = [\mathbf{x};\ \mathbf{1}]$, $\mathbf{x}$ is the input variable, $\mathbf{y}$ is the target (output) variable, $\mathbf{y}^{(\text{ANN})}$ is the predicted output variable obtained from ANN, ϕ_h is the activation function (the basis function) of the input variable, $\mathbf{U}^{\text{ANN}}$ is the transition weight matrix, $\mathbf{W}^{\text{ANN}}$ is the output weight matrix and $\boldsymbol{\eta}$ is an unknown error due to measurement or modeling errors [20, 76]. In the current notations, the bias terms are defined within the weight matrices by augmenting the input variable $\mathbf{x}$ with a unit value [73]. In Eq. 2.13, the target variable is modeled as a linear combination of same type of basis functions (i.e. sigmoid, perceptrons, tanh basis functions) parametrized by $\mathbf{U}^{\text{ANN}}$.

Deep ANN of depth K layers is a neural network architecture of the form:

$$\mathbf{y} \approx \mathbf{y}^{(\mathrm{ANN})} = \mathbf{W}^{\mathrm{ANN}\top}\ \phi_{K-1}\left(\mathbf{U}_{K-1}^{\mathrm{ANN}\ \top}\ \phi_{K-2}\left(\cdots\phi_1\left(\mathbf{U}_1^{\mathrm{ANN}\top}\ \bar{\mathbf{x}}\right)\right)\right), \tag{2.14}$$

where ϕ_k and $\mathbf{U}_k^{\mathrm{ANN}}$ are the element-wise nonlinear function and the weight matrix for the kth layer respectively.

2.3.2 Standard Recurrent Neural Network

Recurrent Neural Network (RNN) is a neural network that has at least one feedback connection in addition to the feedforward connections [108]. The standard form of RNN is a discrete dynamical system of the from [107]:

$$\mathbf{h}_{t+1} = f_h(\mathbf{h}_t, \bar{\mathbf{x}}_{t+1}) = \phi_h(\mathbf{U}^{\mathrm{RNN}\top}\ \mathbf{h}_t + \mathbf{V}^{\mathrm{RNN}\top}\ \bar{\mathbf{x}}_{t+1}) \tag{2.15}$$

$$\mathbf{y}_{t+1}^{(\mathrm{RNN})} = \mathbf{W}^{\mathrm{RNN}\top}\ \mathbf{h}_{t+1} \tag{2.16}$$

where $\bar{\mathbf{x}}_{t+1} = [\mathbf{x}_{t+1};\ \mathbf{1}]$, $\mathbf{x}_{t+1}$ is the input vector at time $t+1$, ϕ_h is the activation function as defined in deep ANN and $\mathbf{U}^{\mathrm{RNN}}, \mathbf{V}^{\mathrm{RNN}}$ and $\mathbf{W}^{\mathrm{RNN}}$ are respectively the transition, input and output weight matrices of standard RNN. In Eq. 2.15, the hidden state $\mathbf{h}_{t+1}$ is estimated based on the corresponding input $\mathbf{x}_{t+1}$ and the hidden state $\mathbf{h}_t$ at the previous time step. This delayed input ($\mathbf{h}_t$) can be thought of as a memory for the artificial system modelled by RNN. The order of the dynamical system expressed by RNN is the number of hidden units i.e. the size of the hidden state vector $\mathbf{h}(t)$ [73]. RNN can approximate state variables of any nonlinear difference equations as a linear combination of hidden state of standard RNN as in Eq. 2.16 supported by the universal approximation theorem:

Theorem 2.3.1 (Universal Approximation Theorem)**.** *Any nonlinear dynamical system can be approximated to any accuracy by a recurrent neural network, with no restrictions on the compactness of the state space, provided that the network has enough sigmoidal hidden units [25, 56].*

Similar to other supervised learning methods, ANN and RNN are calibrated

using training data to find the optimal parameters (neuron weights) of the ANN or RNN. Given a set of training sequences:

$$D = \{((\mathbf{x}_1, \mathbf{y}_1)^\ell \cdots (\mathbf{x}_t, \mathbf{y}_t)^\ell \cdots (\mathbf{x}_T, \mathbf{y}_T)^\ell)\}_{\ell=1}^{L},$$

the RNN parameters $\boldsymbol{\theta} = \{\mathbf{U}^{\text{RNN}},\ \mathbf{V}^{\text{RNN}},\ \mathbf{W}^{\text{RNN}}\}$ are fitted by minimizing the function:

$$\mathbf{J}_{\text{MSE}}(\boldsymbol{\theta}) = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{t=1}^{T} \|\mathbf{y}_t - \mathbf{y}_t^{(\text{RNN})}\|_2^2, \tag{2.17}$$

where $\mathbf{J}_{\text{MSE}}$ known as mean square error (MSE) is the average distance between the observed data $\mathbf{y}_t$ and the RNN output $\mathbf{y}_t^{\text{RNN}}$ across a number of samples L with time dependent observations ($t = 1 \cdots T$ and $\ell = 1 \cdots L$) [107]. We use Euclidean norm to measure the distance between $\mathbf{y}_t$ and $\mathbf{y}_t^{(\text{RNN})}$. Euclidean norm, commonly called as 2-norm comes under a class of vector norm called $p-$norm [131, 46] that takes the form

$$\|\mathbf{y}\|_p = \left(\sum_{i=1}^{n} |y_i|^p\right)^{\frac{1}{p}} \qquad (1 \leq p < \infty) \tag{2.18}$$

The set of parameters $\boldsymbol{\theta}$ could be estimated by backpropagating the gradient of the loss function $\mathbf{J}_{\text{MSE}}$ with respect to $\boldsymbol{\theta}$ in time. This technique is commonly called Backpropagation Through Time (BPTT) [138, 123, 108, 100].

Similar to deep learning Neural Network architectures, standard RNN has training difficulties especially in the presence of long-term dependencies due to the vanishing and exploding gradient [108, 100]. The main reason for the vanishing gradient problem is the exponential dependency of the error function gradient with respect to the weight parameters $\boldsymbol{\theta}$ and the repeated multiplication of error function due to the cyclic behaviour of RNN during BPTT. This repeated multiplication causes the gradient to vanish when the absolute values of weight parameters are less than one [108, 100].

2.3.3 Long Term Short Term Memory network

LSTM architecture [75] was introduced to address the aforementioned vanishing gradient problem. The architecture of LSTM is of the form:

$$\begin{aligned} \mathbf{i} &= \sigma(\mathbf{U}_i^{\text{RNN}^\top}\ \mathbf{h}_t + \mathbf{V}_i^{\text{RNN}^\top}\ \bar{\mathbf{x}}_{t+1}) & \mathbf{f} &= \sigma(\mathbf{U}_f^{\text{RNN}^\top}\ \mathbf{h}_t + \mathbf{V}_f^{\text{RNN}^\top}\ \bar{\mathbf{x}}_{t+1}) \\ \mathbf{o} &= \sigma(\mathbf{U}_o^{\text{RNN}^\top}\ \mathbf{h}_t + \mathbf{V}_o^{\text{RNN}^\top}\ \bar{\mathbf{x}}_{t+1}) & \mathbf{g} &= \tanh(\mathbf{U}_g^{\text{RNN}^\top}\ \mathbf{h}_t + \mathbf{V}_g^{\text{RNN}^\top}\ \bar{\mathbf{x}}_{t+1}) \\ \mathbf{c}_{t+1} &= \mathbf{c}_t\ \circ\ \mathbf{f} + \mathbf{g}\ \circ\ \mathbf{i} & \mathbf{h}_{t+1} &= \tanh(\mathbf{c}_{t+1})\ \circ\ \mathbf{o} \end{aligned} \tag{2.19}$$

where $\mathbf{i}, \mathbf{f}, \mathbf{o}$ are the input, forget and output gates respectively, with sigmoid activation functions σ. These activation functions take the same inputs namely $\mathbf{x}_{t+1}, \mathbf{h}_t$ but utilize different weight matrices $\mathbf{U}^{\text{RNN}}, \mathbf{V}^{\text{RNN}}$ as denoted by the different subscripts. As the name implies, $\mathbf{i}, \mathbf{f}$ and $\mathbf{o}$ act as gates to channelize the flow of information in the hidden layer. For example, the activation of gate $\mathbf{i}$ in channelizing the flow of hidden state $\mathbf{g}$ is done by multiplication of $\mathbf{i}$ with the hidden state value $\mathbf{g}$ [94, 39]. Input gate $\mathbf{i}$ and forget gate $\mathbf{f}$ decides the proportion of hidden state's internal memory $\mathbf{c}_t$ and the proportion of $\mathbf{g}$ respectively to update $\mathbf{c}_{t+1}$. Finally, the hidden state $\mathbf{h}_{t+1}$ is computed by the activation of the output gate $\mathbf{o}$ in channelizing flow of internal memory $\mathbf{c}_{t+1}$. If the LSTM has more than one hidden unit then the operator $\circ$ in Eq. 2.19 is an element-wise multiplication operator.

2.4 Physics driven Deep Residual RNN

General nonlinear dynamical systems (as formulated by Eq. 2.1) are often discretized using implicit time integration schemes to allow for large time steps exceeding the numerical stability constraints [116]. For example, the resultant discrete nonliear dynamical system using Euler implicit time integration scheme takes the form

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \Delta t\ \mathbf{A}\ \mathbf{y}_{t+1} + \Delta t\ \mathbf{F}(\mathbf{y}_{t+1}) \tag{2.20}$$

This leads to a system of nonlinear residual equations at each time step t that takes the form:

$$\mathbf{r}_{t+1} = \mathbf{y}_{t+1} - \mathbf{y}_t - \Delta t\ \mathbf{A}\ \mathbf{y}_{t+1} - \Delta t\ \mathbf{F}(\mathbf{y}_{t+1}) \tag{2.21}$$

To be noted, the residual equation of ROM (Eq. 2.5 and Eq. 2.12) takes a similar form to Eq. 2.21. Generally, the resultant system of nonlinear equations is solved at each time step to minimze the residual using Newton's method. Newton method is an iterative method involving many number of iterations in which a matrix equation is solved at every iteration to update the state variable $\mathbf{y}_{t+1}$. Figure 2.2 displays the pictorial representation of the algorithm to solve Eq. (2.20) using Newton's iterative method.

In addition to the computational burden of Newton's iterations, performing parametric uncertainty propagation requires solving a large number of realizations, in which, each forward realization of the model may involve thousands of time steps, therefore, requiring to perform a very large number of nonlinear iterations. To alleviate this computational burden, we introduce a computationally efficient deep RNN architecture which we denote as deep residual recurrent neural network (DR-RNN) to reflect the physics of the dynamical systems.

DR-RNN iteratively minimize the residual equation (Eq. 2.21) at each time step by stacking K network layers. The architecture of DR-RNN is formulated as:

$$\begin{aligned} \mathbf{y}_{t+1}^{(k)} &= \mathbf{y}_{t+1}^{(k-1)} - \mathbf{w}^{\text{RNN}}\ \circ\ \phi_h(\mathbf{U}^{\text{RNN}}\ \mathbf{r}_{t+1}^{(k)}) && \text{for } k = 1, \\ \mathbf{y}_{t+1}^{(k)} &= \mathbf{y}_{t+1}^{(k-1)} - \frac{\eta_k}{\sqrt{G_k + \epsilon}}\ \mathbf{r}_{t+1}^{(k)} && \text{for } k > 1, \end{aligned} \tag{2.22}$$

where $\mathbf{U}^{\text{RNN}}, \mathbf{w}^{\text{RNN}}, \eta_k$ are the training parameters of DR-RNN, ϕ_h is an activation function (tanh in the current study), the operator $\circ$ in Eq. 2.22 denotes an element-wise multiplication operator, $\mathbf{r}_{t+1}^{(k)}$ is the residual in layer k obtained by substituting $\mathbf{y}_{t+1} = \mathbf{y}_{t+1}^{(k-1)}$ into Eq. 2.21 and G_k is an exponentially decaying squared norm of the residual defined as:

$$G_k = \gamma\ \|\mathbf{r}_{t+1}^{(k)}\|^2 + \zeta\ G_{k-1} \tag{2.23}$$

where γ, ζ are fraction factors and ϵ is a smoothing term to avoid divisions by zero.

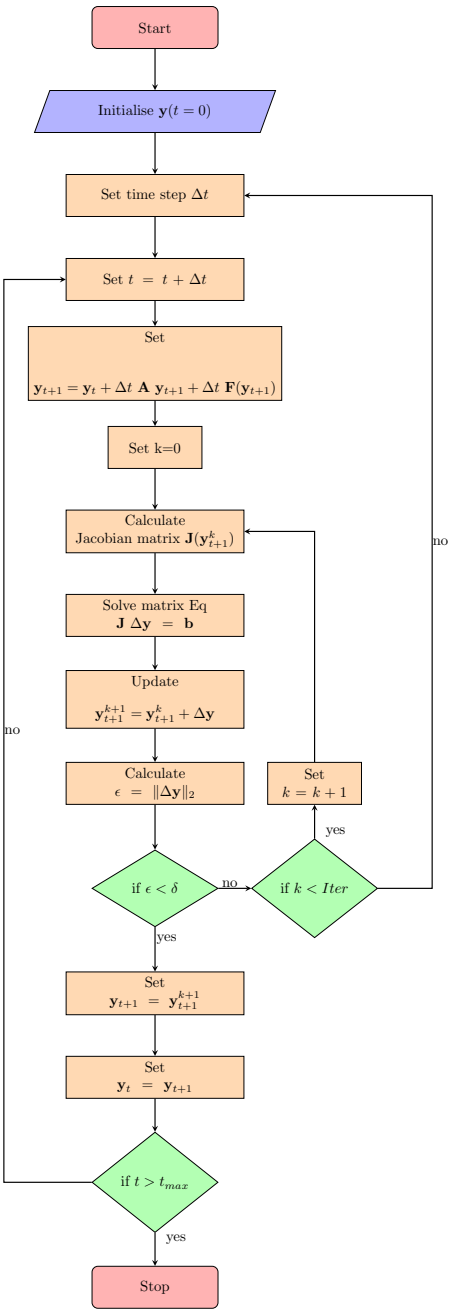


Figure 2.2: Flowchart representation of algorithm utilized to solve Eq. (2.20) using Newton's method.

In this formulation, we set $\mathbf{y}^{(k=0)}_{t+1} = \mathbf{y}_t$. The DR-RNN output at each time step is defined as:

$$\mathbf{y}^{(\mathrm{RNN})}_{t+1} = \mathbf{W}^{\mathrm{RNN}\top}\mathbf{y}^K_{t+1} \tag{2.24}$$

where $\mathbf{W}^{\mathrm{RNN}}$ is a weight matrix that could be optimized during the DR-RNN training process. However, in all our numerical test cases $\mathbf{W}^{\mathrm{RNN}}$, was excluded from the training process and is set as a constant identity matrix. The update equation for $k > 1$ in Eq. 2.22 is inspired by the rmsprop algorithm [130] which is a variant of

the steepest descent method. In rmsprop, the parameters are updated using the following equation:

$$\begin{aligned} G_k &= (1-\gamma)\ (\nabla_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta}^{(k)}))^2 + \gamma\ G_{k-1}, \\ \boldsymbol{\theta}^{(k)} &= \boldsymbol{\theta}^{(k-1)} - \frac{\eta}{\sqrt{G_k+\epsilon}}\ \nabla_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta}^{(k-1)}), \end{aligned} \tag{2.25}$$

where G_k is an exponentially decaying average of the squared gradients of the loss function $\mathbf{J}(\boldsymbol{\theta})$, γ is the fraction factor (usually 0.9), η is the constant learning rate parameter (usually 0.001) and ϵ is a smoothing term to avoid divisions by zero. We note that G_k in Eq. 2.25 is a vector and changes both the step length and the direction of the steepest decent update vector. However, G_k in Eq. 2.22 is a scaler and changes only the step size to update $\mathbf{y}_{t+1}^{(k)}$ in the direction of $\mathbf{r}_{t+1}^{(k)}$. Furthermore, we use G_k as a stability factor in updating $\mathbf{y}_{t+1}^{(k)}$ since the update scheme in DR-RNN is explicit in time and may be prone to instability when using large time steps.

One of the main reasons to consider DR-RNN as a low computational budget numerical emulator is the way the time sequence of the state variables is updated. The dynamics of DR-RNN are explicit in time with a fixed computational budget of order $\mathcal{O}(n)$ per time step. Furthermore, DR-RNN framework has a prospect of applying DR-RNN to solve Eq. 2.21 on different levels of time step much larger than the time step Δt taken in Eq. 2.21. In other words, DR-RNN provides an effective way to solve Eq. 2.21 for a fixed time discretization error.

2.5 Numerical Results

In this section, we demonstrate two different applications of DR-RNN as a model reduction technique. The first application concerns the use of DR-RNN for reducing the computational complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ at each time step for nonlinear ODE systems without reducing the dimension of the state variable of the system. Moreover, DR-RNN is allowed to take large time steps violating the numerical stability condition and is constrained to have time discretization error several times less than the order of large time step taken. We denote this reduction in computational

complexity as temporal model reduction. The second application is focused on spatial dimensionality reduction of dynamical systems governed by a time dependent PDEs with parametric uncertainty. In this case, we use DR-RNN to approximate a reduced order model derived using a POD-Galerkin strategy.

In section 2.5.1, DR-RNN is first demonstrated for temporal model order reduction. In addition, we provide a numerical comparison against ROM based on standard recurrent neural networks architectures. In section 2.5.2, we build DR-RNN to approximate POD based reduce order model (Eq. 2.5) and compare the performance of DR-RNN in approximating POD based ROM against the ROM based on the POD-Galerkin and POD-DEIM methods.

2.5.1 Temporal model reduction

In this section, we conduct temporal model reduction to evaluate the performance of DR-RNN in comparison to the standard recurrent neural networks on three test problems. The standard recurrent neural networks used are RNN and LSTM denoted by RNN_m and LSTM_m respectively, where the subscript m denotes the order of the recurrent neural network (m =number of neurons in the hidden layer). The DR-RNN is denoted by DR-RNN_m, where the subscript m in this case denotes the number of residual layers. We also note that the order of DR-RNN is same as the order of the given dynamical equation since we rely on using the exact expression of the system dynamics. In all test cases, we utilize a tanh activation function in the standard RNN models.

All the numerical evaluations are performed using the keras framework [35], a deep learning python package using Theano [128] library as a backend. Further, we train all RNN models using rmsprop algorithm [130, 35] as implemented in keras with default settings. We set the weight matrix $\mathbf{U}^{\text{RNN}}$ of DR-RNN in Eq. 2.22 as a constant identity matrix and do not include it in the training process. The vector training parameter $\mathbf{w}^{\text{RNN}}$ in Eq. 2.22 is initialized randomly from a zero-mean Gaussian distribution with standard deviation fixed to 0.1. The scalar training parameters η_k in Eq. 2.22 are initialized randomly from the uniform distribution $\mathtt{U}[0.1, 0.4]$. We

set the hyperparameters ζ and γ in Eq. 2.23 to 0.9 and 0.1, respectively.

Problem 1

We consider a nonlinear dynamical system of order $n = 3$ defined by:

$$\begin{aligned} \frac{dy_1}{dt} &= y_1\ y_3, \\ \frac{dy_2}{dt} &= -y_2\ y_3, \\ \frac{dy_3}{dt} &= -y_1^2 + y_2^2 \end{aligned} \tag{2.26}$$

with initial conditions $y_1(0) = 1$, $y_2(0) = 0.1x$, $y_3(0) = 0$. The input x is a random variable with a uniform distribution $\mathtt{U}[-1, 1]$. Modeling this dynamical system is particularly challenging as the response has a discontinuity at the planes $y_1(0) = 0$ and $y_2(0) = 0$ [19]. Figure 2.3 shows the jump discontinuities in the response $y_2(t = 10)$ and $y_3(t = 10)$ versus the perturbations in the initial input x. A standard backward Euler method is used for 100 time steps of size $\Delta t = 0.1$ and we solve the problem for 1500 random samples of x.

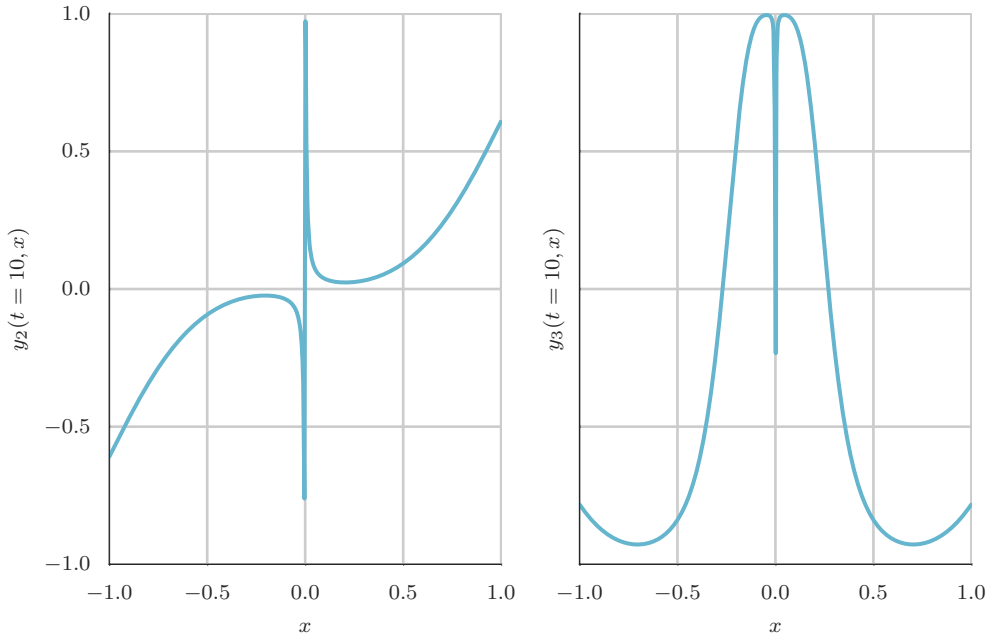


Figure 2.3: System response versus the different values of the initial value random variable x. Note the jump discontinuities in the response $y_2(t = 10)$ and $y_3(t = 10)$ at $x = 0$.

We train RNN parameters using data obtained from 500 random samples and the

remaining runs (i.e. 1000) are used for validation. The training is performed using a batch size of 15 for 15 iterations. We use 7 recurrent neural networks namely RNN_n, RNN_{10n}, LSTM_n, LSTM_{10n}, DR-RNN_1, DR-RNN_2 and DR-RNN_4. The performances of all 7 RNNs are evaluated based on accuracy and model complexity. Accuracy is measured using the mean square error (Eq. 2.17) for the training and the test data sets. Also, we show comparative plots of the probability density function (PDF) of the state variables at specific time steps. Model complexity is determined based on the number of parameters d fitted in each RNN model.

Figure 2.4 compares the PDF of $y_2(t = 10)$ and $y_3(t = 10)$ computed from all 7 RNN against the reference PDF solution. The results presented in Figure 2.4 shows that the PDF obtained from DR-RNN with residual layers closely follow the trend of the reference PDF. The MSE of all RNN models and the corresponding model complexity are presented in Table 2.1. It is worth noticing that DR-RNN models have fewer number of parameters d and hence much lower model complexity than standard RNN models. Furthermore, Table 2.1 shows that DR-RNN with residual layers is considerably better than the standard RNN in fitting the data both in the training and testing data sets. We argue that such performance is due to the iterative update of DR-RNN output towards the desired output. However, the small differences among the models with residual layers indicates that the additional residual layers in DR-RNN_4 are not needed in this particular problem.

Table 2.1: Performance chart of all 7 RNN in problem 1 where d is the number of parameters fitted in RNN and MSE (Eq. 2.17) measures the accuracy of RNN.

Model	RNN_n	RNN_{10n}	LSTM_n	LSTM_{10n}	DR-RNN_1	DR-RNN_2	DR-RNN_4
d	33	84	1093	4053	3	4	6
MSE train	$23{\cdot}10^{-2}$	$15{\cdot}10^{-2}$	$21{\cdot}10^{-2}$	$15\cdot10^{-2}$	$2{\cdot}10^{-3}$	$4{\cdot}10^{-5}$	$4{\cdot}10^{-6}$
MSE test	$23{\cdot}10^{-2}$	$15{\cdot}10^{-2}$	$21{\cdot}10^{-2}$	$14\cdot10^{-2}$	$5{\cdot}10^{-3}$	$4{\cdot}10^{-5}$	$4{\cdot}10^{-6}$

We further train the DR-RNN using data sampled at time interval larger than those used in the backward Euler numerical integrator. For example, we train using sampled data at $\Delta t = 0.5$ resulting in 20 time samples instead of 100 time samples when using the original time step size $\Delta t = 0.1$. We analyse this experiment

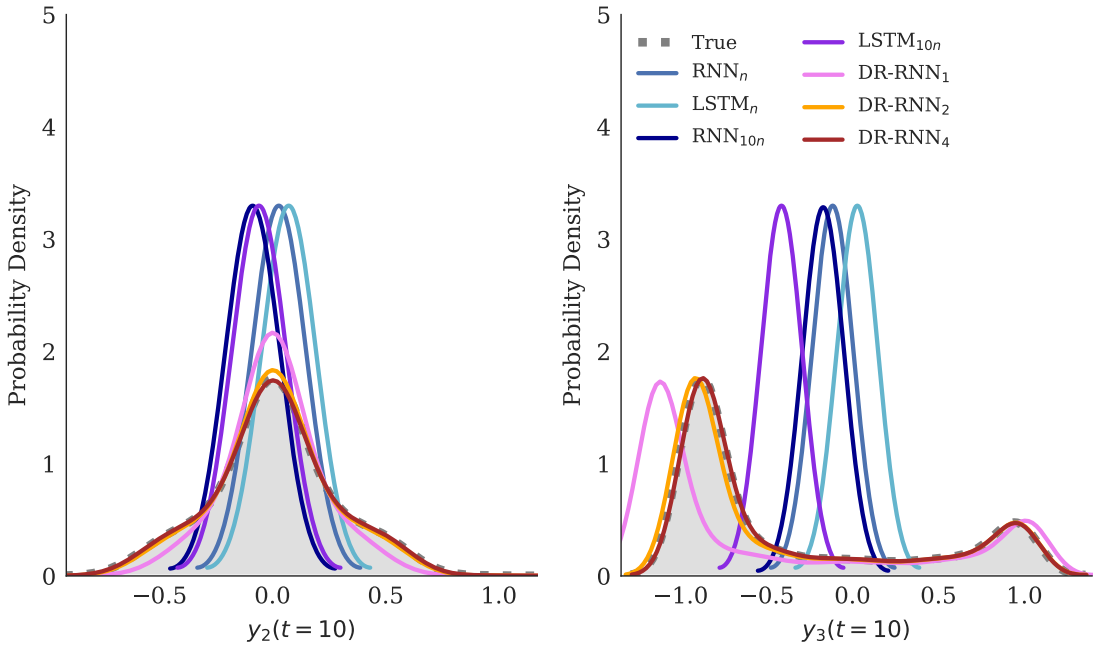


Figure 2.4: Comparison of kernel density estimated probability density function (PDF) of $y_2(t = 10)$ (left) and $y_3(t = 10)$ (right) obtained from all RNN w.r.t. true PDF in problem 1. Label RNN denotes standard RNN (equation 2.15). Subscripts (n or $10n$) in the label RNN and LSTM denotes the dimension of hidden layer where n is the dimension of state variable $\mathbf{y}$. Subscript in the label DR-RNN denotes the number of layers K in DR-RNN. The dimension of all the layers in all DR-RNN is n.

using DR-RNN$_2$ and DR-RNN$_4$ as the top performer in the last set of numerical experiments. Figure 2.5 shows the PDF of $y_3(t = 10)$ computed from DR-RNN$_2$ and DR-RNN$_4$ for different time step along with the PDF computed from the reference solution. As can be observed, the performance of DR-RNN$_4$ is superior to DR-RNN$_2$ supporting our argument on the hierarchical iterative update of the DR-RNN solution as the number of residual layer increases. In Figure 2.5, DR-RNN$_2$ performed well for 2 times $\Delta t = 0.1$, while it results in large errors for 5 and 10 times $\Delta t = 0.1$ whereas DR-RNN$_4$ performed well for all large time steps. Through this numerical experiment, we provide numerical evidence that DR-RNN is numerically stable when approximating the discrete model of the true dynamical system for a range of large time steps with small discretization errors. However, there is a limit on the time step size for the desired accuracy in the output of the DR-RNN and this limit is correlated to the number of utilized layers.

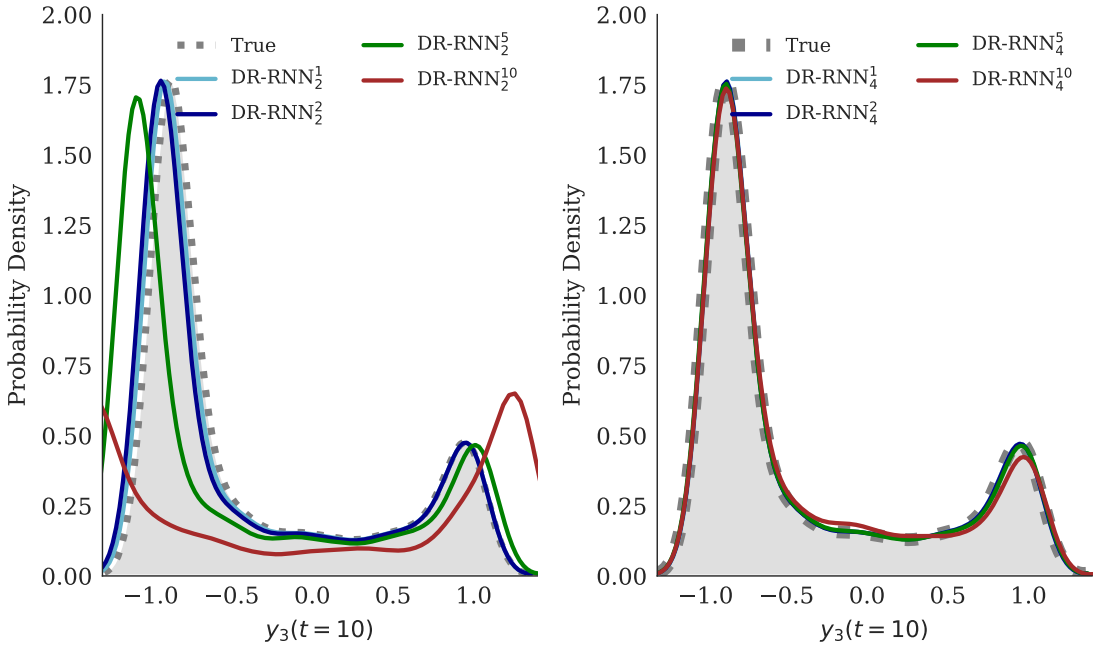


Figure 2.5: Comparison of kernel density estimated probability density function (PDF) of $y_3(t = 10)$ (left) and $y_3(t = 10)$ (right) obtained from DR-RNN for different large time step size w.r.t. true PDF computed from fine step size in problem 1. Subscript in the label DR-RNN denotes the number of layers K in DR-RNN. Superscript in the label DR-RNN denotes how many times the large time step bigger than the fine step size.

Problem 2

The dynamical equation for problem 2 is the same as in test problem 1. However, the initial conditions are set to $y_1(0) = 1,\ y_2(0) = 0.1\ x_1,\ y_3(0) = x_2$ where the stochastic dimension is increased from 1 to 2. The input random variables x_1, x_2 are modeled by uniform probability distribution function $\mathtt{U}[-1, 1]$. We adopted the same procedure followed in problem 1 to evaluate the performances of the proposed DR-RNN in-comparison to the standard recurrent neural network models. Figure 2.6 shows a comparison of the PDF plot for $y_2(t = 10)$ and $y_3(t = 10)$ computed from all RNN models. Errors of all RNN models and the corresponding model complexity are presented in Table 2.2. We can see the performance trend of all RNN models observed in problem 2 are similar to the trends observed in Problem 1.

We follow the similar procedure adopted in problem 1 to analyze the performance of DR-RNN in taking large time step. Figure 2.7 compares the PDF of $y_3(t = 10)$ computed from DR-RNN$_2$ and DR-RNN$_4$ for different large time steps with the PDF

Table 2.2: Performance chart of all 7 RNN in problem 2 where d is the number of parameters fitted in RNN and MSE (Eq. 2.17) measures the accuracy of RNN.

Model	RNN_n	RNN_{10n}	LSTM_n	LSTM_{10n}	DR-RNN_1	DR-RNN_2	DR-RNN_4
d	33	84	1093	4053	3	4	6
MSE train	$26{\cdot}11^{-2}$	$26{\cdot}10^{-2}$	$26{\cdot}10^{-2}$	$20\cdot10^{-2}$	$2{\cdot}10^{-2}$	$1{\cdot}10^{-4}$	$2{\cdot}10^{-6}$
MSE test	$26{\cdot}11^{-2}$	$26{\cdot}10^{-2}$	$26{\cdot}10^{-2}$	$20\cdot10^{-2}$	$2{\cdot}10^{-2}$	$1{\cdot}10^{-4}$	$3{\cdot}10^{-6}$

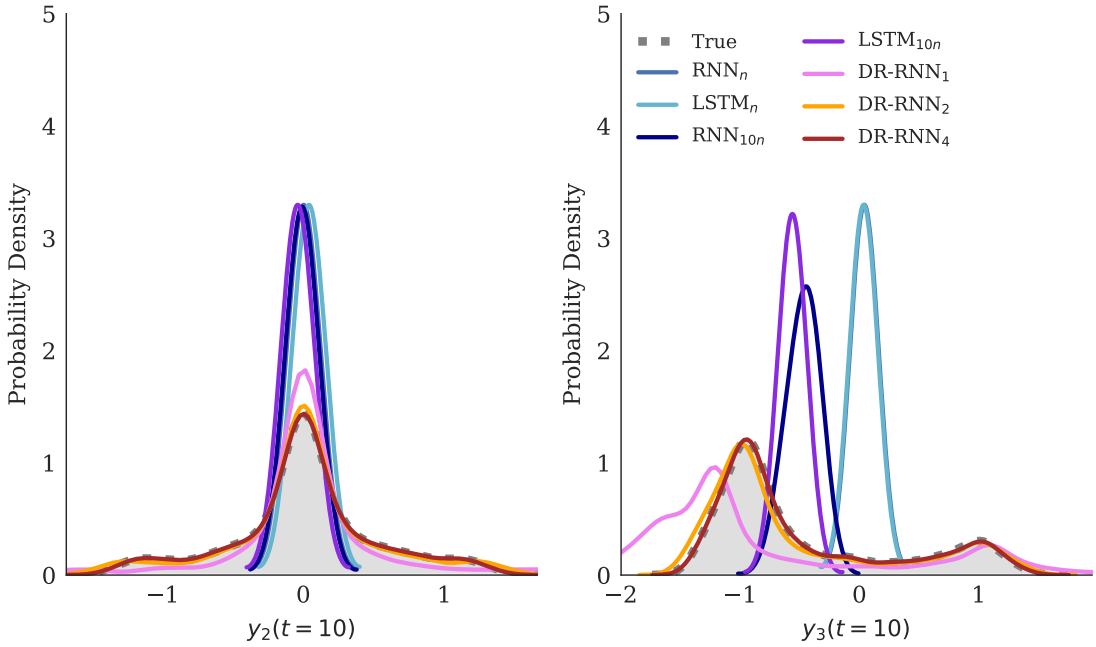


Figure 2.6: Comparison of kernel density estimated probability density function (PDF) of $y_2(t=10)$ (left) and $y_3(t=10)$ (right) obtained from all RNN w.r.t. true PDF in problem 2. Label RNN denotes standard RNN (equation 2.15). Subscripts (n or $10n$) in the label RNN and LSTM denotes the dimension of the hidden layer where n is the dimension of state variable $\mathbf{y}$. Subscript in the label DR-RNN denotes the number of layers K in DR-RNN. The dimension of all the layers in all DR-RNN is n.

computed from the reference solution for the fine time step size. We observe similar performance trends of DR-RNN_2 and DR-RNN_4 to those observed in test problem 1 (Figure 2.5).

Problem 3

The dynamical system considered in this problem is similar to problem 1 and problem 2 with further additional difficulties in the initial conditions $y_1(0) = x_1,\ y_2(0) =$

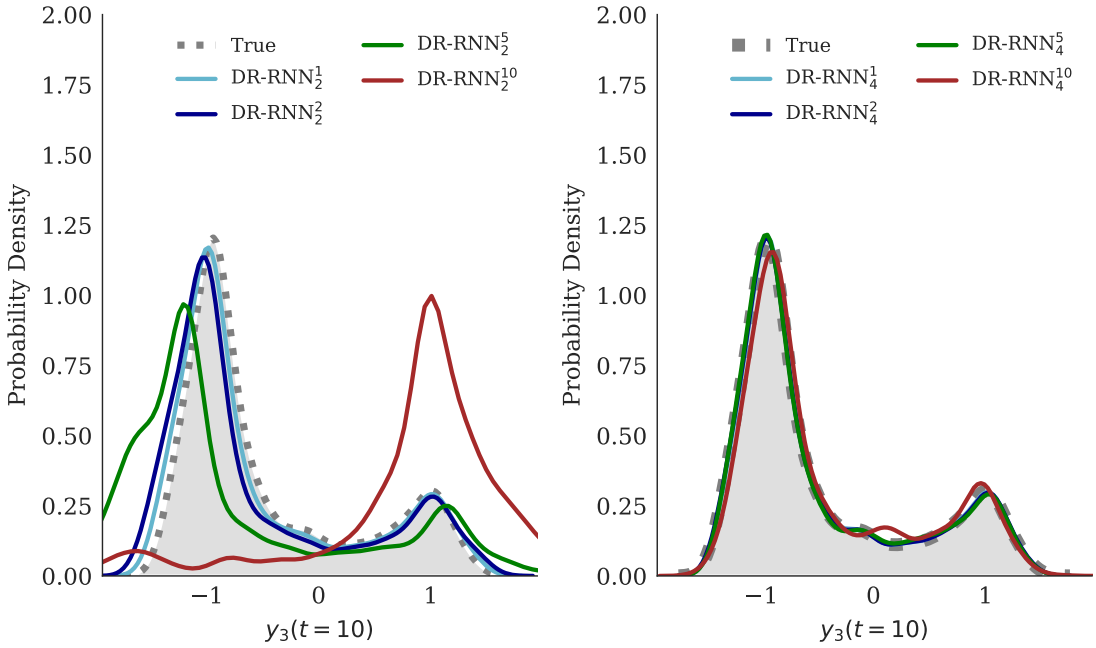


Figure 2.7: Comparison of kernel density estimated probability density function (PDF) of $y_3(t = 10)$ (left) and $y_3(t = 10)$ (right) obtained from DR-RNN for different large time step size w.r.t. true PDF computed from fine step size in problem 2. Subscript in the label DR-RNN denotes the number of layers K in DR-RNN. Superscript in the label DR-RNN denotes how many times the large time step bigger than the fine step size.

x_2, $y_3(0) = x_3$, where $x_1, x_2, x_3 \in \mathtt{U}[-1, 1]$. Remarkably, problem 3 is rather difficult to train by RNN compared to problem 1 as the stochastic dimension in this problem is 3. We adopted the same procedure followed in problem 1 to evaluate the performances of the proposed DR-RNN in comparison to the standard recurrent neural network models. Figure 2.8 shows the PDF of $y_2(t = 10)$ and $y_3(t = 10)$ computed from all RNN. Errors of all RNN models and their model complexity are presented in Table 2.3. Performance ranking of all 7 RNN models remain similar to Problem 1 and Problem 2 in spite of the increased stochastic dimension. More specifically, from Table 2.3, we notice a decreases in MSE as the number of network layers in DR-RNN increases.

We carry out the same large time step performance analysis carried out in problem 1 and problem 2 for DR-RNN$_2$ and DR-RNN$_4$. Figure 2.9 compares the PDF of $y_3(t = 10)$ using DR-RNN$_2$ and DR-RNN$_4$ for different large time step with the PDF computed from the reference solution using the fine time step size. One can

Table 2.3: Performance chart of all 7 RNN in problem 3 where d is the number of RNN parameters and MSE (Eq. 2.17) measures the accuracy of RNN.

Model	RNN_n	RNN_{10n}	LSTM_n	LSTM_{10n}	DR-RNN_1	DR-RNN_2	DR-RNN_4
d	33	84	1093	4053	3	4	6
MSE train	$33{\cdot}10^{-2}$	$17{\cdot}10^{-2}$	$33{\cdot}10^{-2}$	$15\cdot10^{-2}$	$3{\cdot}10^{-3}$	$1{\cdot}10^{-4}$	$1{\cdot}10^{-6}$
MSE test	$33{\cdot}10^{-2}$	$17{\cdot}10^{-2}$	$33{\cdot}10^{-2}$	$15\cdot10^{-2}$	$4{\cdot}10^{-2}$	$5{\cdot}10^{-4}$	$1{\cdot}10^{-6}$

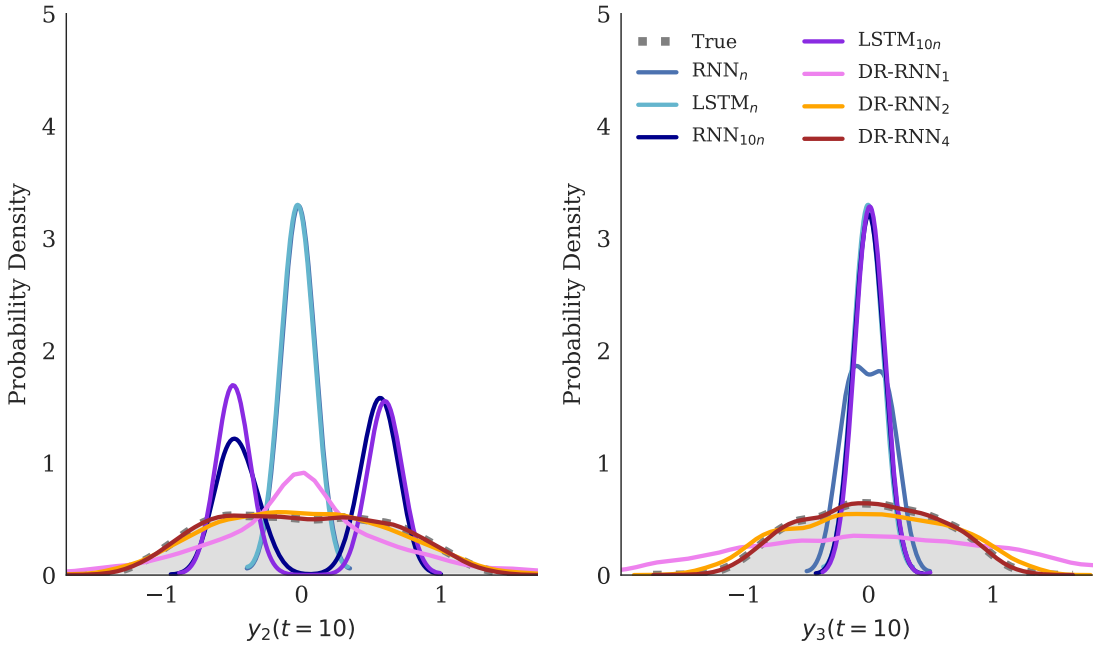


Figure 2.8: Comparison of kernel density estimated probability density function (PDF) of $y_2(t=10)$ (left) and $y_3(t=10)$ (right) obtained from all RNN w.r.t. true PDF in problem 3. Label RNN denotes standard RNN (Eq. 2.15). Subscripts (n or $10n$) in the label RNN and LSTM denotes the dimension of the hidden layer where n is the dimension of state variable $\mathbf{y}$. Subscript in the label DR-RNN denotes the number of output layers K in DR-RNN. The dimension of all the layers in all DR-RNN is n.

notice the performance trend of DR-RNN_2 and DR-RNN_4 are nearly similar to the trend noticed in problem 1 and problem 2 (Figure 2.5 and Figure 2.7). From the results presented in Figure 2.9, we observe that DR-RNN_4 performs well for large time steps of 2, 5 times $\Delta t = 0.1$, however, it results in small errors in the PDF plot for the case of 10 times $\Delta t = 0.1$ in this problem.

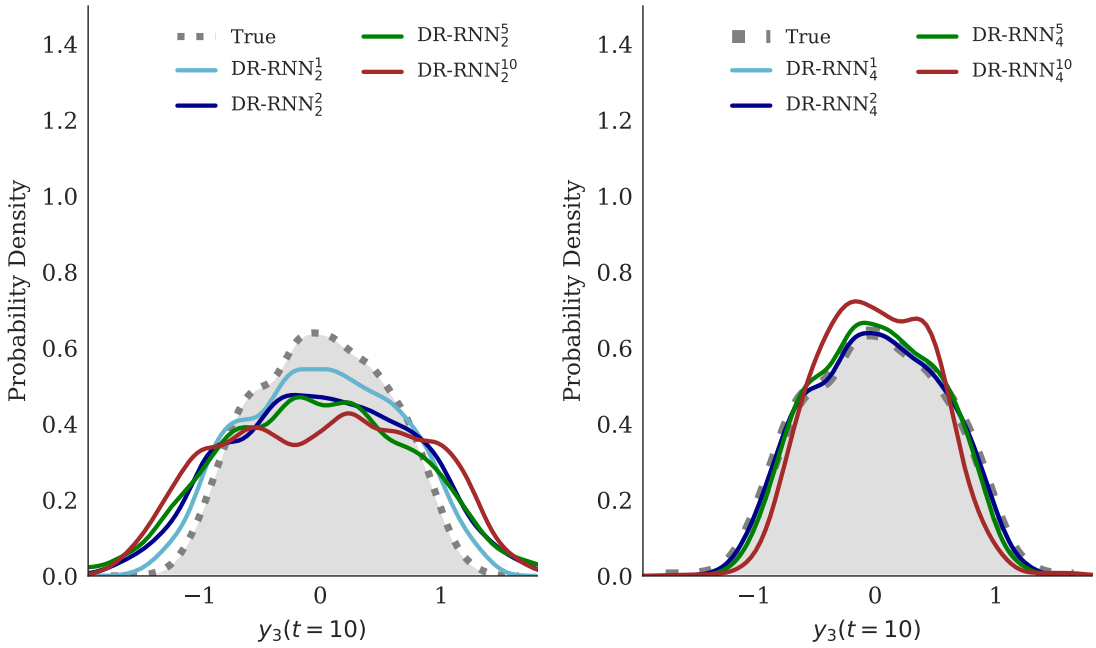


Figure 2.9: Comparison of kernel density estimated probability density function (PDF) of $y_3(t = 10)$ (left) and $y_3(t = 10)$ (right) obtained from DR-RNN computed for different large time step size w.r.t. true PDF computed from fine step size in problem 3. Subscript in the label DR-RNN denotes the number of layers K in DR-RNN. Superscript in the label DR-RNN denotes how many times the large time step bigger than the fine step size.

2.5.2 Dimensionality reduction in space

In this section, we evaluate the performance of DR-RNN in spatial dimensional reduction by using DR-RNN to approximate the ROM derived from POD-Galerkin strategy. We compare DR-RNN with POD based ROM and POD-DEIM ROM to conduct two parametric uncertainty quantification problems involving time dependent partial differential equations.

In the following test cases, the weight matrix $\mathbf{U}^{\text{RNN}}$ of DR-RNN in Eq. 2.22 is initialized randomly from a uniform distribution function $\mathtt{U}[0.1, 0.5]$. The vector training parameter $\mathbf{w}^{\text{RNN}}$ in Eq. 2.22 is initialized randomly from the white Gaussian distribution with its standard deviation fixed to 0.1. The scalar training parameters η_k in Eq. 2.22 are initialized randomly from the uniform distribution $\mathtt{U}[0.1, 0.4]$. We set the hyperparameters ζ, γ in Eq. 2.23 to $0.9, 0.1$ respectively.

Problem 4

In this problem we model unsteady heat diffusion over the spatial domain $x = [0, 1]$ using:

$$\frac{\partial \mathbf{y}}{\partial t} = -\alpha \, \frac{\partial^2 \mathbf{y}}{\partial x^2} + \mathbf{g} \tag{2.27}$$

where $\mathbf{y}$ is the temperature field, α is the random heat diffusion coefficient with uniform probability distribution function $\mathtt{U}[0.01, 0.08]$. The problem is imposed with homogeneous initial condition $\mathbf{y}(x, 0) = 0$ and Dirchlet boundary conditions $y(0, t) = 0$ and $y(1, t) = 0$. The heat source $\mathbf{g}$ takes the form:

$$\mathbf{g} = \begin{cases} 1 & \text{if } x \in [0.4, 0.6] \\ \\ 0 & \text{else} \end{cases} \tag{2.28}$$

We use a finite difference discretization with a spatial step size $\Delta x = 0.01$. The discretized FOM is formulated as:

$$\frac{d\mathbf{y}}{dt} = \mathbf{A} \; \mathbf{y} + \mathbf{b} \tag{2.29}$$

with $\mathbf{A} \in \mathcal{R}^{n \times n}$ obtained using second order central difference stencil. The dimension of the problem is $n = 99$. The resulting system of ODEs (Eq. 2.29) is then discretized in time using standard implicit Euler method and the resultant time discrete system takes the form

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \Delta t \; \mathbf{A} \; \mathbf{y}_{t+1} + \Delta t \; \mathbf{b} \tag{2.30}$$

We solve FOM (Eq. (2.30) for 40 time steps with a time step size $\Delta t = 0.03$. We solve the problem for 500 random samples of α. Further, a set of solution snapshots is collected to construct the POD basis by computing the following singular value decomposition

$$\mathbf{X} = \mathbf{U} \; \Sigma \; \mathbf{W}^* \qquad \mathbf{U} \in \mathcal{R}^{n \times n} \qquad \Sigma \in \mathcal{R}^{n \times N_s} \qquad \mathbf{W} \in \mathcal{R}^{N_s \times N_s} \tag{2.31}$$

where $\mathbf{X}$ is the snapshot matrix of the sample solutions of Eq. 2.29, N_s is the number of snapshots used in computing SVD. The space of $\mathbf{y}$ is spanned by the orthonormal column vectors of matrix $\mathbf{U}$. The left panel in the Figure 2.10 shows the decay of singular values of the snapshot matrix $\mathbf{X}$. The optimal basis for approximating $\mathbf{y}(t)$ is given by the first r columns of matrix $\mathbf{U}$ denoted by $\mathbf{U}_r$ and is used to reduce the FOM given by Eq. 2.29 to POD based ROM of the form:

$$\frac{d\tilde{\mathbf{y}}}{dt} = \tilde{\mathbf{A}}\ \tilde{\mathbf{y}} + \tilde{\mathbf{b}} \tag{2.32}$$

where $\tilde{\mathbf{A}} = \mathbf{U}_r^\top\ \mathbf{A}\ \mathbf{U}_r$ and $\tilde{\mathbf{b}} = \mathbf{U}_r^\top\ \mathbf{b}$. Next, we solve Eq. 2.32 using standard implicit Euler method with a time step of size $\Delta t = 0.03$ for 40 time steps using the same 500 random samples of α used in FOM (Eq. 2.29). We solve Eq. 2.32 for a set of different number of POD basis functions ($r = 2, 4, 5, 7, 15$). Finally, we built DR-RNN with four layers to approximate the ROM defined in Eq. 2.32. We train DR-RNN using time snapshot solutions of Eq. 2.32 collected for some random samples of heat diffusion coefficient.

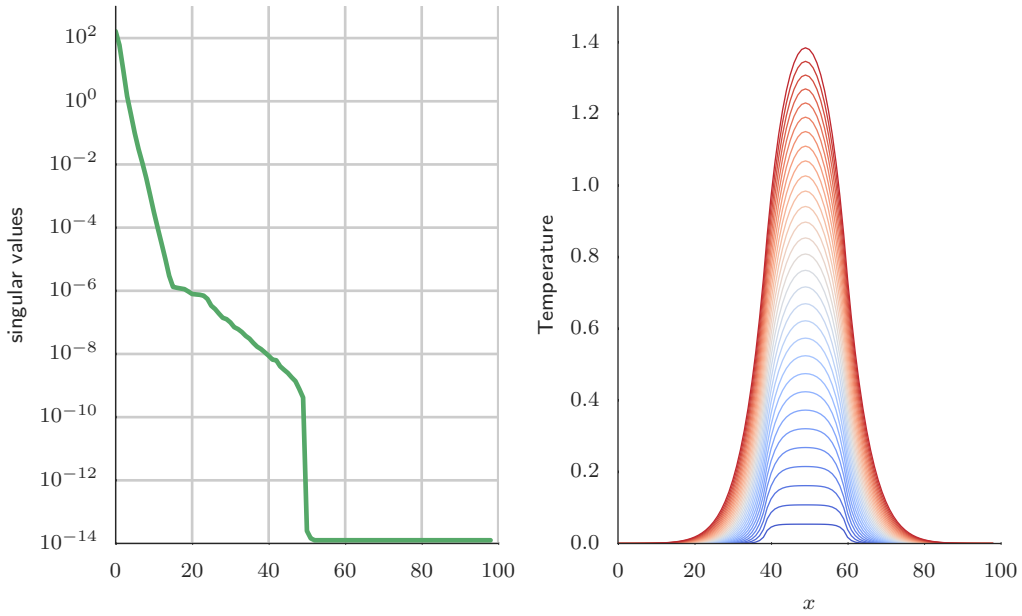


Figure 2.10: Left: Singular values of the solution snapshot matrix $\mathbf{X}$. Right: Numerical Solutions of the full-order system $n = 99$ in problem 4.

Figures 2.10 and 2.11 show the numerical solutions obtained from the FOM, the linear POD of dimension 15, and the DR-RNN of dimension 15. The results plotted in the figures show that both the POD based ROM and the DR-RNN with

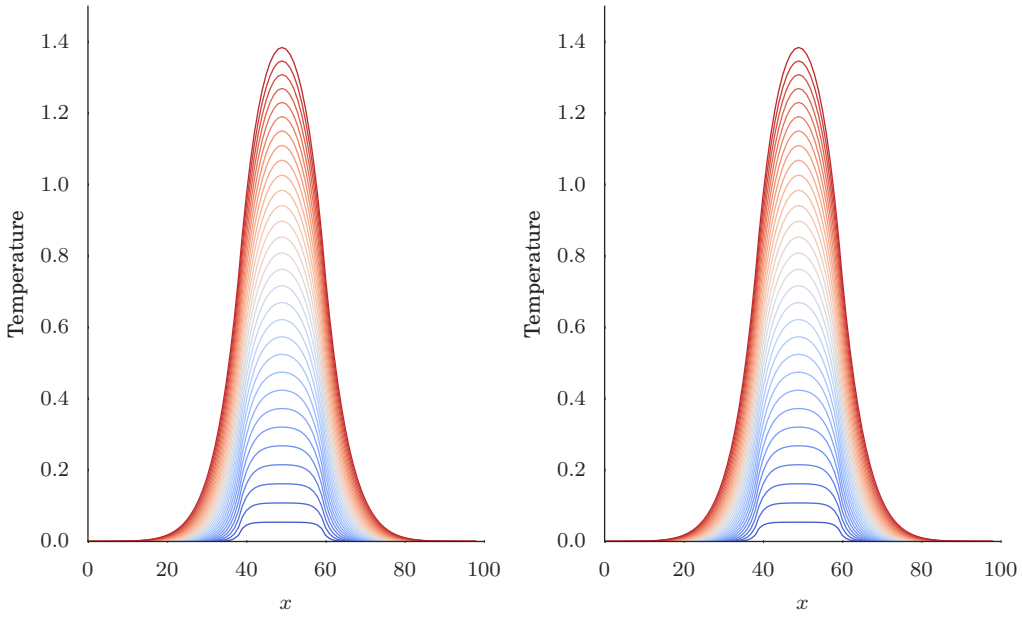


Figure 2.11: Numerical solutions for problem 4 at different time steps. Left: POD-Galerkin reduced system with 15 POD basis. Right: DR-RNN using 15 POD basis. Dimension of the full-order model $n = 99$.

dimension 15 produce good approximations to the original full-order system.

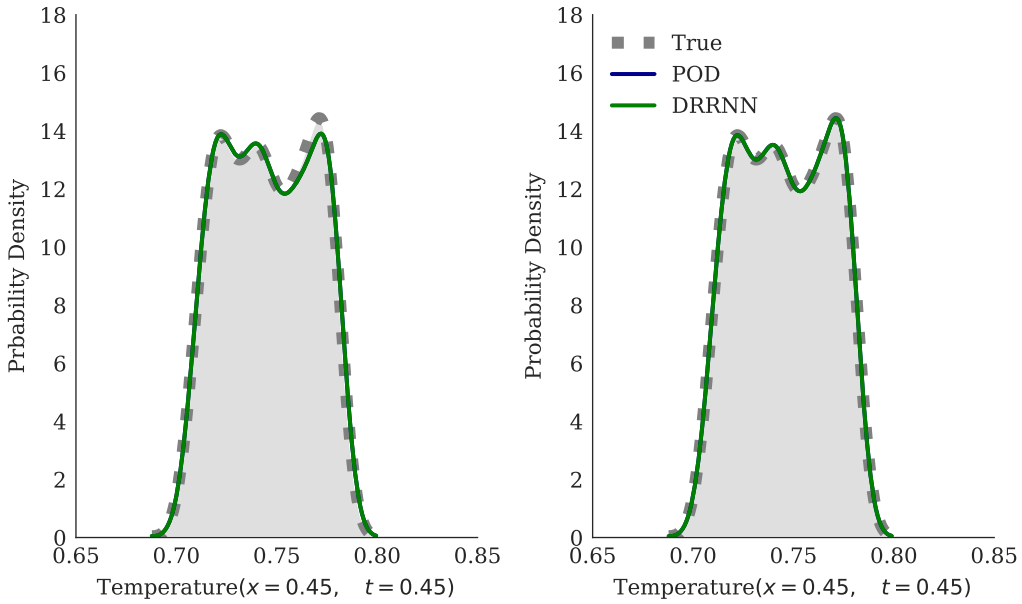


Figure 2.12: Comparison of kernel density estimated probability density function (PDF) obtained from POD ROM, and DR-RNN w.r.t. true PDF obtained from full-order system in problem 4. Left: number of POD basis used = 5. Right: number of POD basis used = 15. Dimension of the full-order model $n = 99$.

Figure 2.12 compare PDF of $\mathbf{y}(x = 0.45, t = 0.45)$ obtained from the reduced order models against the full order model PDF. The utilized ROMs use 5 POD basis functions in the left panel and 15 POD basis functions in the right panel.

The results in Figure 2.12 shows that the PDF obtained by the reduced systems are indistinguishable from the PDF of the FOM, while using 5 or 15 POD basis. Figure 2.13 shows the MSE defined in Eq. 2.17 for different number of POD basis obtained from the POD based ROM and the DR-RNN. From the Figure 2.13, we can observe that the MSE decreases with the increase in the number of POD basis due to the decay of singular values of the snapshot solution matrix $\mathbf{X_s}$. Although the results of DR-RNN and POD based ROM are indistinguishable, we note that DR-RNN is an explicit method with a computational complexity of $\mathcal{O}(T \times L \times r^2)$ while POD method uses an implicit time discretization scheme with a complexity nearly to $\mathcal{O}(T \times L \times r^3)$, where T is the number of time steps marched in the time domain and L is the number of random samples.

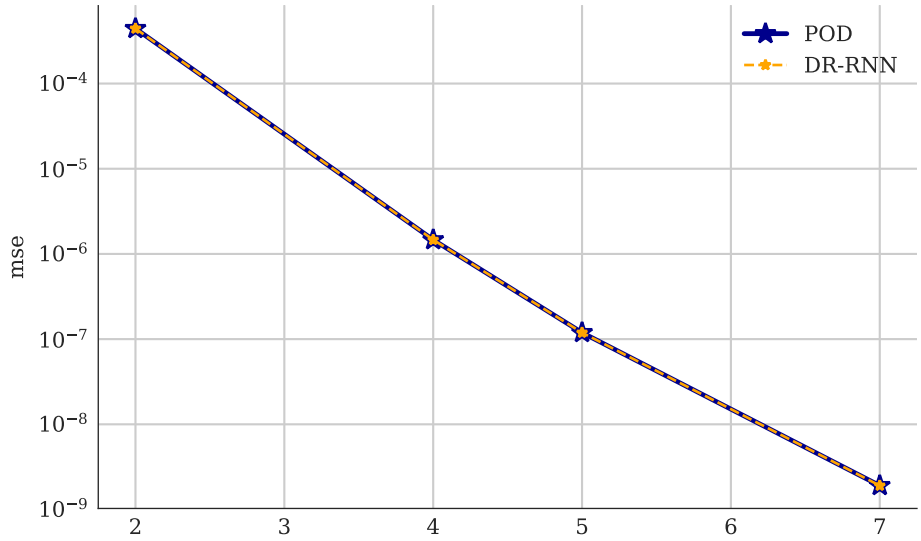


Figure 2.13: Comparison of MSE defined in Eq. 2.17 obtained from POD and DR-RNN ROM in problem 4.

Problem 5

In this problem, we are interested in modeling the fluid displacement within a porous media, where water is pumped to displace oil. Although the displacing fluid is assumed to be immiscible with the displaced fluid (oil), the displacement front does not take place as a piston like flow process with a sharp interface between the two fluids. Rather, simultaneous flow of the immiscible fluids takes place within the

porous media [34]. In this problem, we are mainly interested in the evolution of the saturation of the water phase. We solve a pair of partial differential equations namely the pressure and the saturation equations. A simplified one-dimensional pressure equation takes the form [34]:

$$\frac{\partial}{\partial x}\left(\lambda \mathbf{K} \cdot \frac{\partial \mathbf{p}}{\partial x}\right) + \mathbf{q} = 0 \tag{2.33}$$

where $\mathbf{p}$ is the pressure, $\mathbf{K}$ is the permeability, λ is the total mobility, $\mathbf{q}$ is the mass flow rate defined as $\mathbf{q} = \mathbf{q}_w/\rho_w + \mathbf{q}_o/\rho_o$ and ρ is the density. The subscript w and o denotes the water phase and the oil phase, respectively. The mobility term is defined as $\lambda = \lambda_w + \lambda_o$, where

$$\lambda_w = \frac{k_{rw}}{\mu_w} \qquad \lambda_o = \frac{k_{ro}}{\mu_o},$$

μ is the viscosity and k_{rw}, k_{ro} are the relative permeability terms defined by the Brooks-Corey model [34, 1]. The second equation is the saturation equation defined as [34]:

$$\phi \frac{\partial \mathbf{s}}{\partial t} + \frac{\partial \left(\mathbf{v} \cdot f_s\right)}{\partial x} + \frac{\mathbf{q}_w}{\rho_w} = 0 \tag{2.34}$$

where $\mathbf{v} = -(\lambda_w + \lambda_o)\ \mathbf{K}\ (\partial \mathbf{p}/\partial x)$ is the Darcy velocity field, ϕ is the porosity and $f_s = \lambda_w/(\lambda_w + \lambda_o)$ is the fractional flow function. We complete the model description by defining the phase relative permeabilities as a function of saturation using Brooks-Corey model [34]:

$$k_{rw} = s^{*2} \qquad k_{ro} = (1 - s^*)^2 \qquad s^* = s - s_{or} - s_{ow} \tag{2.35}$$

where s_{or} is the residual oil saturation, s_{ow} is the residual water saturation and s is the saturation value. In this simplified model, we assume a constant porosity throughout the media and we neglect the effects of compressibility, capillary, and

gravity. We complete the description of the problem by the following input data:

$$\begin{aligned} \mathbf{q}(x=0) &= 0.1 \qquad \mathbf{q}(x=1) = -0.1 \\ \mu_w &= 0.1 \qquad \mu_o = 1 \\ s_{or} &= 0.2 \qquad s_{ow} = 0.2 \end{aligned} \tag{2.36}$$

The initial condition of $\mathbf{s}$ is uniform and is equal to s_{ow} and we use no flow boundary condition. We adopt a sequantial implicit solution strategy [34, 1] to compute the numerical solution of Eq. 2.33 and Eq. 2.34. In this method, a sequential updating of the velocity field and saturation is performed where each equation is treated separately. The first step is to solve for the pressure and the velocity field at an initial time. Then, with this velocity field and initial saturation, the saturation is evolved over a small number of time steps with the velocity field kept constant. The resulting saturation is then used to update the pressure and velocity. The process is repeated until the time of interest. We use a simple finite volume method (FVM) for spatial discretization with first order upwind scheme as it is a conservative method. The discretized form of the FOM of Eq. 2.34 is formulated as:

$$\frac{d\mathbf{s}}{dt} = \mathbf{A}\ \mathbf{f}(\mathbf{s}) + \mathbf{b} \tag{2.37}$$

This equation is then discretized in time using backward Euler method to a time discrete system that takes the form The resulting system of ODEs (Eq. 2.29) is then discretized in time using standard implicit Euler method and the resultant time discrete system takes the form

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \Delta t\ \mathbf{A}\ \cdot \mathbf{f}(\mathbf{s}^{t+1}) + \Delta t\ \mathbf{b} \tag{2.38}$$

In space, we use 64 spatial grid points over the domain $x = [0, 1]$ and in time we use a time step of size $\Delta t = 0.015$ for 100 time steps. Newton Raphson iteration is used to solve the resulting system of nonlinear equations to evolve the saturation at each time step. The uncertainty parameter in this test problem is the porosity value ϕ with a uniform probability distribution function $\mathtt{U}[\mathtt{0.18}, \mathtt{0.38}]$. We solve the

FOM (Eq. 2.37) by using standard implicit Euler method for 500 random samples of ϕ. It is interesting fact to note that constant $\Delta t = 0.03$ violates Von Neumann stability condition given by

$$\Delta t \leq \frac{\phi \cdot \Delta x}{max\left(\mathbf{v} \cdot \frac{df}{ds}\right)} \tag{2.39}$$

where Δx is numerical grid size [116, 129, 1]. Next, the POD basis vectors are constructed from the solutions of the full-order system taken from the collected set of snapshot solutions. This is done by computing the following singular value decomposition

$$\begin{aligned} \mathbf{X_s} &= \mathbf{U}\ \Sigma_\mathbf{s}\ \mathbf{W_s^*} \qquad \mathbf{U} \in \mathcal{R}^{n\times n} \qquad \Sigma_\mathbf{s} \in \mathcal{R}^{n\times N_s} \qquad \mathbf{W_s} \in \mathcal{R}^{N_s\times N_s} \\ \mathbf{X_f} &= \mathbf{V}\ \Sigma_\mathbf{f}\ \mathbf{W_f^*} \qquad \mathbf{V} \in \mathcal{R}^{n\times n} \qquad \Sigma_\mathbf{f} \in \mathcal{R}^{n\times N_f} \qquad \mathbf{W_f} \in \mathcal{R}^{N_f\times N_f} \end{aligned} \tag{2.40}$$

where $\mathbf{X_s}$ is the snapshot matrix of saturation and $\mathbf{X_f}$ is the snapshot matrix of nonlinear function $\mathbf{f}(\mathbf{s})$, $n = 64$ is the dimension of $\mathbf{s}$ and N_s, N_f are the number of snapshots used in computing SVD for the saturation and the nonlinear flow function respectively. The space of saturation is spanned by the orthonormal column vectors of matrix $\mathbf{U}$ and the space of nonlinear function $\mathbf{f}(\mathbf{s})$ is spanned by the orthonormal column vectors in the matrix $\mathbf{V}$. The optimal basis for approximating $\mathbf{s}(t)$ is given by the first r columns of the matrix $\mathbf{U}$ denoted by $\mathbf{U}_r$ and is used to reduce FOM to POD based ROM of the form:

$$\frac{d\tilde{\mathbf{s}}}{dt} = \mathbf{U}_r^\top\ \mathbf{A}\ \mathbf{f}(\mathbf{U}_r\ \tilde{\mathbf{s}}) + \tilde{\mathbf{b}} \tag{2.41}$$

where $\mathbf{s} \approx \mathbf{U}_r\ \tilde{\mathbf{s}}$, $\tilde{\mathbf{b}} = \mathbf{U}_r^\top\ \mathbf{b}$ and $\mathbf{U}_r^\top\ \mathbf{A}\ \mathbf{f}(\mathbf{U}_r\ \tilde{\mathbf{s}})$ forms the bottleneck that has to be reduced with DEIM as detailed in Section 2.2.2. Application of the DEIM algorithm (section 2.2.2) on nonlinear POD based ROM (Eq. 2.41) results in POD-DEIM reduced order model of the form:

$$\frac{d\tilde{\mathbf{s}}}{dt} = \tilde{\mathbf{A}}\ \mathbf{f}(\mathbf{P}^\top\mathbf{U}_r\ \tilde{\mathbf{s}}) + \tilde{\mathbf{b}} \tag{2.42}$$

where $\tilde{\mathbf{A}} = \mathbf{U}_r^\top \mathbf{A}\ \mathbf{D}$, $\mathbf{D} = \mathbf{V}_m(\mathbf{P}^\top\mathbf{V}_m)^{-1}$ referred as DEIM-matrix in Eq. 2.9 (section 2.2.2), $\mathbf{V}_m$ is the orthogonal matrix for optimally approximating $\mathbf{f}(\mathbf{s})$ given by the first m columns of the matrix $\mathbf{V}$. Figure 2.14 shows the decay of singular values of the snapshot matrix $\mathbf{X_s}$ and of the nonlinear function snapshot matrix $\mathbf{X_f}$. Next, we solve Eq. 2.41 and Eq. 2.42 by using standard implicit Euler method with a time step of $\Delta t = 0.03$ for 100 time steps using the same 500 random samples of ϕ used in FOM (Eq. 2.37). We solve Eq. 2.41 for a set of POD basis functions ($r = 15, 35, 55$) and similarly, we solve Eq. 2.42 for the same set of POD basis functions using a DEIM basis functions of fixed number ($m = 35$). Further, we built DR-RNN to approximate the POD-DEIM ROM (Eq. 2.42) where we apply DEIM in the DR-RNN to evaluate the nonlinearity, which gives an important speedup in the efficiency of the formulation. We train DR-RNN using time snapshot solutions of Eq. 2.41 collected for some random samples of porosity values.

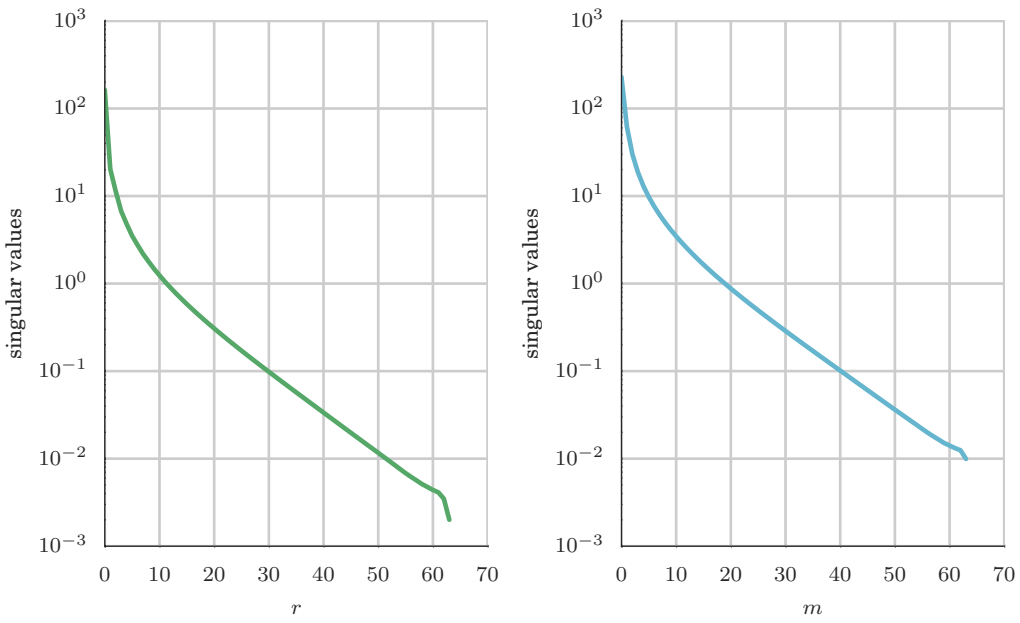


Figure 2.14: Left: Singular values of the solution snapshot matrix $\mathbf{X_s}$. Right: Singular values of the nonlinear function snapshot matrix $\mathbf{X_f}$.

Figure 2.15 compares the kernel density estimated probability density function (PDF) obtained from all ROMs to the PDF obtained from the FOM. Figure 2.16 compares the numerical solutions obtained from all the reduced order models to the numerical solutions obtained from the FOM. In these figures, ROM uses 15 POD basis functions in the left panel and 35 POD basis functions in the right panel. From

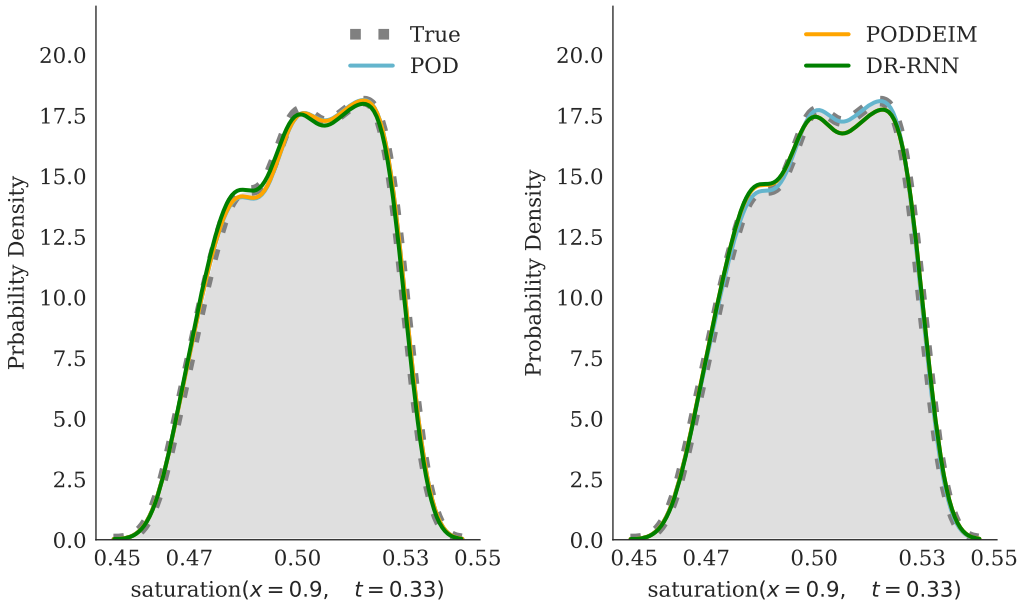


Figure 2.15: Comparison of kernel density estimated probability density function (PDF) obtained from all ROMs w.r.t. true PDF obtained from full-order system in problem 5. Left: number of POD basis used = 15. Right: number of POD basis used = 35. Dimension of the full-order model $n = 64$.

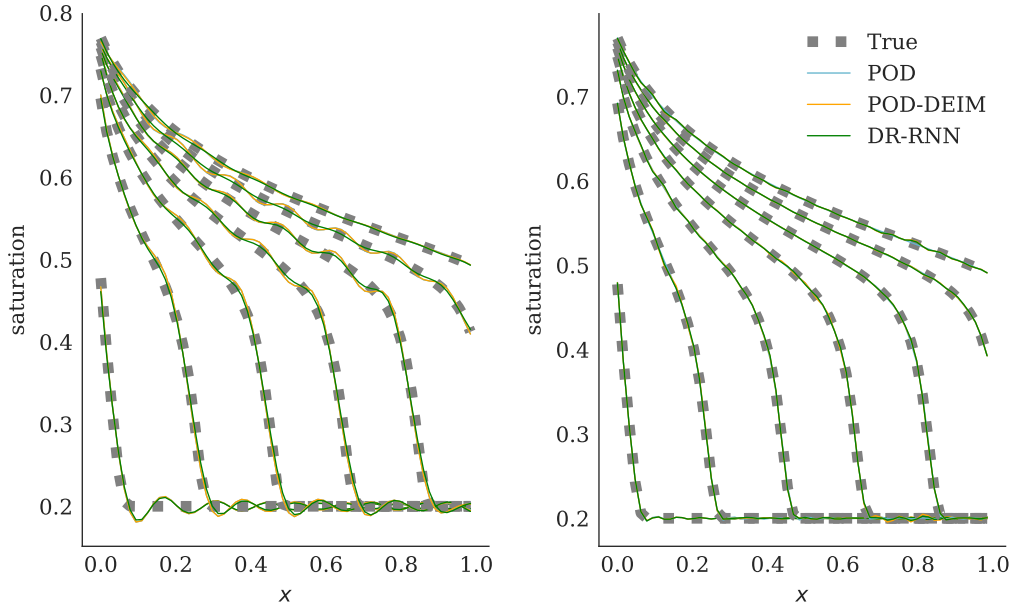


Figure 2.16: Numerical solution of the saturation equation obtained from all ROMs w.r.t. full-order system in problem 5. Left: number of POD basis used = 15. Right: number of POD basis used = 35. Dimension of the full-order model $n = 64$ and porosity value used $\phi = 0.2$.

these figures, when the POD basis of dimension 35 is used, the numerical solutions of the reduced systems from all approaches appear to be indistinguishable from the numerical solution of the FOM. We note that the saturation equation has a hyperbolic

structure which is more complicated to capture, especially in the nonlinear function. The difficulty due to hyberbolic structure of the saturation equation in constructing POD ROM system is more pronounced when POD ROM is constructed using less number of basis functions. For example, in this problem, we observe numerical oscillations in the saturations profile obtained from POD based ROMs when using 15 basis functions (left of Figure 2.16) whereas POD ROM using 35 basis functions can accurately approximate the solutions of the FOM saturation profile (right of Figure 2.16). Figure 2.17 shows the MSE defined in Eq. 2.17 at different

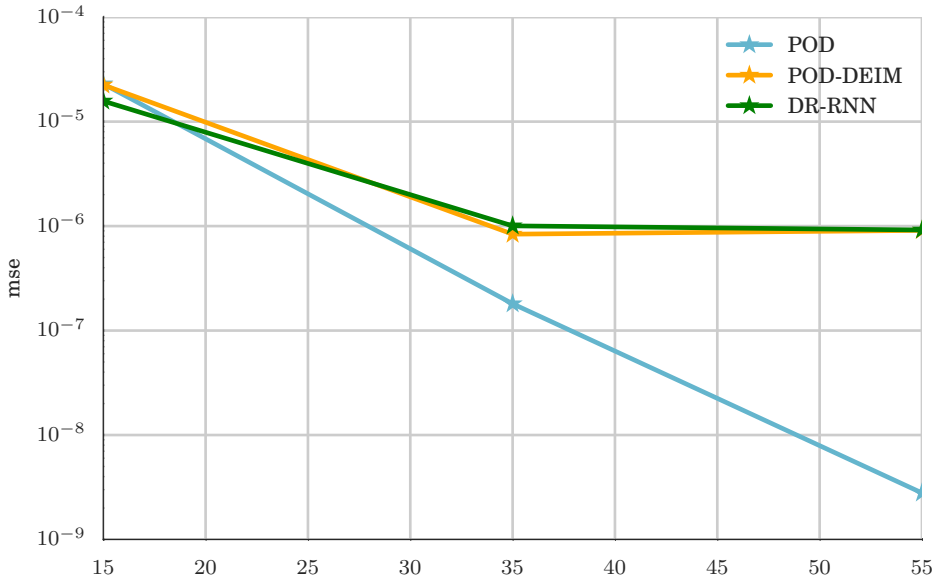


Figure 2.17: Comparison of MSE defined in Eq. 2.17 obtained from all ROMs in problem 5.

number of POD basis obtained from all ROMs. From Figure 2.17, we can observe a decrease in MSE as we increase the number of POD basis which is attributed to the decay of singular values of the snapshot solution matrix $\mathbf{X_s}$. Although the errors from the POD reduced system is slightly lower than the errors arising from applying POD-DEIM and DR-RNN, the complexity in the on-line computation of the nonlinear function $\mathbf{f}(\mathbf{s})$ still depends on the dimension of the original full-order system. Moreover, it is necessary to compute the Jacobian matrix of full dimension at every Newton iteration and at every time step in POD based ROM [30]. Despite the fact that POD-DEIM approach not only gives an accurate reduced system with reduced computational complexity by removing the dependency on the dimension

of the original full-order system with the general nonlinearities, POD-DEIM relies on evaluating the Jacobian matrix at every Newton iteration and results in a computational complexity in order $\mathcal{O}(T \times L \times p \times r^3)$, where p is the number of Newton iterations. The presented numerical results showed that both POD-DEIM and DR-RNN approaches can be used to construct an accurate reduced system. However, DR-RNN constructs an accurate reduced system without evaluating the Jacobian matrix (as an explicit method) and thus limiting the computational complexity to $\mathcal{O}(T \times L \times K \times r^2)$ instead of $\mathcal{O}(T \times L \times p \times r^3)$, where $K \ll p$ is the number of stacked network layers and p is the number of Newton iterations.

2.6 Conclusions

In this chapter, we introduced a Deep Residual Recurrent Neural Network (DR-RNN) as an efficient model reduction technique that accounts for the dynamics of the full order physical system. We then presented a new model order reduction for nonlinear dynamical systems using DR-RNN in combination with POD based MOR techniques. We demonstrated two different applications of the developed DR-RNN to reduce the computational complexity of the full-order system in different computational examples involving parametric uncertainty quantification. Our first application concerned the use of DR-RNN for reducing the computational complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ for nonlinear ODE systems. In this context, we evaluate DR-RNN in emulating nonlinear dynamical systems using a different number of large time step sizes violating the numerical stability condition for a small time discretization errors. The presented results showed an increased accuracy of DR-RNN as the number of residual layer increases based on the hierarchical iterative update scheme.

The second application of DR-RNN is related to spatial dimensionality reduction of dynamical systems governed by time dependent PDEs with parametric uncertainty. In this context, we used DR-RNN to approximate ROM derived from a POD-Galerkin strategy. For the nonlinear case, we combined POD with the DEIM algorithm for approximating the nonlinear function. The developed DR-RNN pro-

vides a significant reduction of the computational complexity of the extracted ROM limiting the computational complexity to $\mathcal{O}(K \times r^2)$ instead of $\mathcal{O}(p \times r^3)$ per time step for the nonlinear POD-DEIM method, where $K \ll p$ is the number of stacked network layers in DR-RNN and p is the number of Newton iterations in POD-EIM. This chapter has illustrated the applicability of DR-RNN as en effective ROM for an one dimensional porous media flow problem with uncertainty in the scalar porosity value. The next chapter will consider the application of DR-RNN as an effective ROM in an UQ problem involving two dimensional porous media flow with a high dimensional uncertainty permeability field.

Chapter 3

Reduced order modeling of subsurface multi-phase flow models using deep residual recurrent neural networks [1]

3.1 Introduction

Simulation of multi-phase flow in a subsurface porous media is an essential task for a number of engineering applications including ground water management, contaminant transport, and effective extraction of hydrocarbon resources [115, 49]. The physics governing subsurface flow simulations are mainly modeled by a system of coupled nonlinear partial differential equations (PDEs) parametrized by subsurface properties (e.g. porosity and permeability) [1]. In realistic settings, subsurface models are computationally expensive (i.e. large number of grid block is needed) as the subsurface properties are heterogeneous and the solution exhibit multiscale features [48, 115].

Moreover, these subsurface properties are only known at a sparse set of points (i.e. well locations) and the grid properties are populated stochastically over the entire domain [78, 48, 49]. Monte-Carlo methods are usually employed to propagate

[1] The contents of this chapter has been published in TIPM journal.

the uncertainties in the subsurface properties to the flow response. Monte-Carlo methods are computationally very expensive since a large number of forward simulations are necessary to estimate the statistics of the engineering quantities of interest [115, 49, 78]. Likewise, Bayesian inference tasks require a very large number of forward simulations to sharpen our knowledge about the unknown model parameters by utilizing field observation data [48, 49]. For example, Markov-chain Monte-Carlo (MCMC) method (and its variants) requires a large number (in millions) of reservoir simulations to reach convergence and to avoid biased posterior estimates of the model parameters.

Surrogate models can be used to overcome the computational burden of multi-query tasks (e.g. uncertainty quantification, model based optimization) governed by large scale PDEs [53, 87, 68, 50, 83, 16]. Surrogate models are computationally efficient mathematical models that can effectively approximate the main characteristics of the full-order model (full model) [53]. A number of surrogate modeling techniques have been developed and could be broadly classified into three classes: simplified physics based models [43, 83], data-fit black-box models [53, 90, 145], and projection based reduced order models commonly referred to as reduced model [17, 89, 5, 52]. Physics based surrogate models are derived from high-fidelity models using approaches such as simplifying physics assumptions, using coarse grids, and/or upscaling of the model parameters [43, 53, 68, 9]. Data-fit models are generated using the detailed simulation data to regress the relation between the input and the corresponding output of interest [53, 145, 2, 8]. For a complete review of various surrogate modeling techniques, we refer the readers to the following papers by Asher et al. [6], Frangos et al. [53], Koziel and Leifsson [87], Razavi et al. [118].

In projection based reduced order models (utilized in this paper), the governing equations of the full model are projected into a low-dimensional subspace spanned by a small set of basis functions via Galerkin projection [89, 5]. Projection based ROMs relies on the assumption that most of the information and characteristics of the full model state variables can be efficiently represented by linear combinations of only a small number of basis functions. This assumption enables reduced

model to accurately capture the input-output relationship of the full model with a significantly lower number of unknowns [53, 89, 5]. Projection based reduced order models are broadly categorized into: system based methods and snapshot based methods. System based methods like balanced truncation realization methods [66], and Krylov subspace methods [54] use the characteristics of the full model and have been developed mainly for linear time-invariant problems, although much progress has been done on extensions of these methods to nonlinear problems [88]. Snapshot based methods such as reduced basis methods [121], proper orthogonal decomposition (POD) [125, 17] derive the projection bases from a set of full model solutions (the snapshots).

In this work, we employ POD based reduced model to accelerate Monte-Carlo simulation of subsurface flow models. The basis functions obtained from the POD is optimal in the sense that, for the same number of basis functions, no other bases can represent the given snapshot set with lower least–squares error than the POD bases [89, 125] (see section 3.3 for further details). Lumley [97] was the first to apply POD techniques in fluid flow simulations. Since then, POD procedures has successfully been applied in a number of application areas [e.g., 125, 148, 26, 23, 99, 7, 82].

In the context of fluid flow in porous media, Vermeulen et al. [134] introduced POD in the confined, groundwater flow problems (linear subsurface flow model). Vermeulen et al. [135] applied POD in gradient-based optimization problem involving groundwater flow model. McPhee and Yeh [98] employed POD to enhance the groundwater management optimization problem. Siade et al. [124] introduced a new methodology for the optimal selection of snapshots in such a way that the resulting POD basis functions account for the maximal variance of the full model solution. Within the context of oil reservoir simulation, Heijn et al. [71] and Van Doren et al. [133] applied POD to accelerate the optimization of a waterflood process. Cardoso et al. [28] incorporated a new snapshot clustering procedure to enhance the standard POD for oilwater subsurface flow problems.

In the context of Monte-Carlo simulations applied to stochastic subsurface flow

problems, POD based ROMs were mainly employed only when the governing equation was linear (or nearly linear) [27, 109, 110, 21]. Pasetto et al. [109] employed POD based reduced model to construct MC realizations of two dimensional steady state confined groundwater flow subject to a spatially distributed random recharge. Pasetto et al. [110] applied POD to accelerate the MC simulations of transient confined groundwater flow models with stochastic hydraulic conductivity. Baú [15] derived a set of POD ROMs for each MC realization of hydraulic conductivity to solve a stochastic, multi-objective, confined groundwater management problem. Boyce and Yeh [21] applied a single parameter-independent POD reduced model to the deterministic inverse problem and the Bayesian inverse problem involving linear groundwater flow model. In addition to the limitation of using only linear flow models, the UQ tasks in the aforementioned literature involve only low dimensional uncertain parameters.

Within the context of nonlinear subsurface flow problems, the target application of POD was mainly hydrocarbon production optimization, where POD ROMs were used mainly to optimize well control parameters (e.g., bottomhole pressure) [27, 69, 132, 120, 81]. Recently, Jansen and Durlofsky [81] has done an extensive review on the use of reduced-order models in well control optimization. For the well control applications, POD achieved reasonable levels of accuracy only when the well controls in test runs were relatively close to those used in training runs. In the case where the test controls substantially differ from those used in the initial training runs, additional computational steps were needed. For example refitting the POD basis functions was performed in [132], which impose some additional computational overhead. Although POD combined with Galerkin projection has been applied more frequently to nonlinear flow problems [23, 17, 120], the effectiveness of POD-Galerkin based model in handling nonlinear systems is limited mainly by two factors. The first factor is related to the treatment of the nonlinear terms in the POD-Galerkin reduced model [30, 119, 27] and the second factor is related to maintaining the overall stability of the resulting reduced model [27, 67, 68, 24, 137].

In relation to computing reduced non-polynomial nonlinear functions, POD

based ROMs is usually dependent on the full model state variables and henceforth, the computational cost of evaluating the reduced model is still a function of full model dimension. Several techniques have been developed to reduce the computational cost of evaluating the nonlinear term in POD ROMs including trajectory piecewise linearization (TPWL) [119], gappy POD technique [140], missing point estimation (MPE) [12], best point interpolation method [106], and discrete empirical interpolation method (DEIM) [12, 30]. Among these techniques, TPWL and DEIM are widely used for efficient treatment of nonlinearities in multi-phase flow reservoir simulations [61, 67, 68].

In TPWL method [119], the nonlinear function is first approximated by a piecewise linear function obtained by linearizing the full-order model at a predetermined set of points in the time and the parameter space. Then the nonlinear full model is replaced by an adequately weighted sum of the selected linearized systems [119]. Finally, the reduced model can be obtained by projecting the resultant linearized full-order system using standard techniques like POD [119]. The TPWL method was first introduced in [119] for modeling nonlinear circuits and micromachined devices. In the context of subsurface flow problems, TPWL procedures were applied in [27, 69, 132, 120] to accelerate the solution of production optimization problems.

In DEIM, the nonlinear term in the full model is approximated by a linear combination of a set of basis vectors [30]. The coefficients of expansion are determined by evaluating the nonlinear term only at a small number of selected interpolation points [30]. DEIM was developed in [30] for model reduction of general nonlinear system of ordinary differential equations (ODEs) and have been used in several areas [32, 142, 22]. Within the context of subsurface flow problems, Chaturantabut and Sorensen [31] applied DEIM for model reduction of viscous fingering problems of an incompressible fluid through a two dimensional homogeneous porous medium. Alghareeb and Williams [4] combined DEIM with POD procedures and the resultant reduced model was applied in waterflood optimization problem. Recently, Ghasemi [61] applied POD with DEIM to an optimal control problem governed by two-phase flow in a porous media. Next, Ghasemi [61] used machine learning technique to con-

struct a number of POD-DEIM local reduced-order models. In that work, machine learning technique was used to construct a number of POD-DEIM local reduced-order models and then a specific local reduced-order model was selected with respect to the current state of the dynamical system during the gradient based optimization task. Similarly, Yoon et al. [146] used multiple local DEIM approximations in POD reduced model framework to reduce the computational costs of high-fidelity reservoir simulations.

The overall convergence and stability is another issue that limits the applicability of POD-Galerkin based ROMs. POD-Galerkin projection methods manage to decrease the computational complexity by orders of magnitude as a result of state variable's dimension reduction. However, this reduction goes hand in hand with a loss in accuracy. Moreover, slow convergence and in some cases model instabilities [137, 67, 24] are observed as the errors in the reduced state variables are propagated in time. More specifically, the performance of POD-Galerkin ROMs is directly influenced by the number of POD basis used in the POD-Galerkin projection. However, in many applications involving nonlinear conservation laws (e.g. high Reynold number fluid flow), POD-Galerkin reduced order models have shown poor performance even after retaining a sufficient number of POD basis [137, 125, 17].

Several stabilization techniques have been proposed in the recent literature to build a stabilized POD based reduced models. A notable stabilization technique relies on closing the POD reduced model using a set of closure models similar to those adopted in turbulence modeling [17, 137]. The objective of applying closure models within POD based reduced model is to include the effects of the discarded POD basis functions in the extracted reduced model [17, 137]. Wang et al. [137] showed that POD-Galerkin reduced model yielded inaccurate and physically implausible results when applied to the numerical simulation of a 3D turbulent flow past a cylinder at Reynold number of 1000. Wang et al. [137] addressed the aforementioned accuracy and stability issues of POD reduced model by various closure models, where artificial viscosity was added to the real viscosity parameter to stabilize the POD based reduced model.

Another major approach to enhance the stability of POD-Galerkin reduced model is to compute a new set of optimal basis or to improve the POD basis vectors by solving a constrained optimization problem. Bui-Thanh et al. [24] determined a new set of optimal basis vectors by formulating an optimization problem constrained by the equations of the resultant reduced model and demonstrated the stability of the proposed approach on linear dynamical systems. We note that POD-Galerkin reduced model orthogonally projects the nonlinear residual into the subspace spanned by the POD basis vectors. Unlike POD-Galerkin reduced model, Petrov-Galerkin projection scheme design a different set of orthonormal basis called left reduced order basis into which the nonlinear residual is projected. Carlberg et al. [29] formulated stable Petrov-Galerkin reduced model in which the left reduced order basis vectors were computed from an optimization problem at every iteration of the Gauss Newton method. He [67] observed that poor spectral properties of the reduced Jacobian matrix could cause numerical instabilities in POD-Galerkin TPWL reduced model. Hence, He [67] improved the stability of the POD based reduced model by determining the optimal dimension of the reduced model through an extensive search over a range of integer numbers. We note that all the above mentioned optimization procedures involve computationally expensive procedures to maintain stability and in many cases, the stability of the extracted reduced model is still not guaranteed [67, 68].

Recently, data-fit black-box models have been combined with POD [143] to develop non-intrusive POD based ROMs, where the data-fit models are used to regress the relationship between the input parameter and the reduced representation of the full model state vector. Hence, non-intrusive ROMs do not require any knowledge of the full-order model and is mainly developed to circumvent the shortcomings in accessing the governing equations of the full model [143]. However, it can also be used to address the stability and nonlinearity issues of POD based ROMs. Wang et al. [136] developed a non-intrusive POD reduced model using Recurrent Neural Network (RNN) as a data-fit model and presented two fluid dynamics test cases namely, flow past a cylinder and a simplified wind driven ocean gyre. RNN is a

class of artificial neural network [108, 100] which has at least one feedback connection in addition to the feedforward connections [107, 108, 79]. In the context of data-fit models, RNN has been successfully applied to various sequence modeling tasks such as automatic speech recognition and system identification of time series data [73, 70, 74, 65]. Additionally, RNN has been applied to emulate the evolution of nonlinear dynamical systems in a number of applications [150, 11] and henceforth has large potential in building reduced order models. However, the applicability of non-intrusive ROMs is severely undermined in many real-world problems, where increasing the dimensionality of the input parameter space increases the complexity and training time of the data-fit model.

In summary, among many surrogate modeling techniques, POD-Galerkin reduced model is a viable option for accelerating multi-query tasks like UQ. Generally, POD-Galerkin reduced model is well established for linear systems and for nonlinear systems with parametric dependence, POD could be either combined with TPWL or with DEIM for modeling subsurface flow systems [27, 69, 132, 61]. However, POD reduced model does not preserve the stability properties of the corresponding full order model and current state of the art POD stabilization techniques [137, 67, 68] are not cost effective and ultimately do not guarantee stability of the extracted reduced order models.

In this work, we use DR-RNN [104] to alleviate the potential limitations of POD-Galerkin reduced models. More specifically, we combine DR-RNN with POD-Galerkin and DEIM methods to derive an accurate and computationally effective reduced model for uncertainty quantification (UQ) tasks. The architecture of DR-RNN is inspired by the iterative line search methods where the parameters of the DR-RNN are optimized such that the residual of the numerically discretized PDEs is minimized [18, 130, 104]. Unlike the standard RNN which is very generic, DR-RNN [104] uses the residual of the discretized differential equation. In addition, the parameters of the DR-RNN are fitted such that the computed DR-RNN output optimally minimizes the residual of the targeted equation. In this context, DR-RNN is a physics aware RNN as it is tailored to leverage the physics embedded in the

targeted dynamical system (i.e. residual of the equation or reduced residual in the current manuscript).

The resultant reduced model obtained from DR-RNN combined with POD-Galerkin and DEIM algorithm has a number of salient features. First, the dynamics of DR-RNN is explicit in time with superior convergence and stability properties for large time steps that violate the numerical stability conditions [104, 116]. Second, as the dynamics modeled in DR-RNN are explicit in time, there is a reduction in the computational complexity of the extracted reduced model from $\mathcal{O}(r^3)$ corresponding to implicit POD-DEIM reduced order models, to $\mathcal{O}(r^2)$, where r is the size of the reduced model. Third, DR-RNN requires only very few training samples (obtained by solving the full model) to optimize the parameters of the DR-RNN as it accounts for the physics of the full model within the RNN architecture (via the reduced residual). This is a major advantage when compared to pure data-driven algorithms (e.g. standard RNN architectures). Moreover, DR-RNN can effectively emulate the parameterized nonlinear dynamical system with a significantly lower number of parameters in comparison to standard RNN architectures [104].

In this work, we demonstrate the superior properties of DR-RNN in accelerating UQ tasks for subsurface reservoir models using Monte-Carlo method. As far as we are aware, the use of a single parameter-independent POD-Galerkin reduced model in Monte-Carlo method involving nonlinear subsurface flow with high dimensional stochastic permeability field has not been previously explored. The reason is that the resultant reduced model might require significantly more basis functions to reconstruct stable solutions [27, 69, 21, 61]. However, only a single set of small number of POD basis functions would be sufficient to reconstruct the solution with reasonable accuracy using least–squares (see section 3.3.2 for more details). Hence, the aim of this study is to illustrate how DR-RNN could be used to reconstruct stable solutions emulating the full model dynamics using only a small set of POD basis functions. The proposed DR-RNN technique is validated on two forward uncertainty quantification problems involving two-phase flow in subsurface porous media. The two flow problems are commonly known within the reservoir simulation community

as the quarter five spot problem and the uniform flow problem [1]. In these two numerical examples, the permeability field is modeled as log-normal distribution. The obtained results demonstrate that DR-RNN combined with POD-DEIM provides an accurate and stable reduced order model with a drastic reduction in the computational cost. The reason for selecting simplified flow problems is to illustrate the potential benefit of DR-RNN to formulate an accurate and computationally effective POD-DEIM reduced model for flow problems where the standard POD-Galerkin reduced models are inaccurate and possibly unstable. We also note that DR-RNN architecture is generic and could be used to emulate any well-posed nonlinear dynamical system [104] including subsurface flow problems while accounting for capillary pressure effects, gravity effects and compressibility.

The outline of the rest of this chapter is as follows: In section 3.2, we present the formulation of multi-phase flow problem in a porous media. In section 3.3, we introduce POD-Galerkin method for model reduction followed by a discussion of DEIM for handling nonlinear systems. In Section 3.4, we describe the architecture of DR-RNN and in section 3.5, we evaluate the reduced model derived by combining DR-RNN with POD-DEIM on two uncertainty quantification test cases. Finally, in Section **??**, we present the conclusions of this manuscript.

3.2 Problem Formulation

The equations governing two-phase flow of a wetting phase (water) and non-wetting phase (e.g. oil) in a porous media are the conservation of mass (continuity) equation and Darcy's law for each phase [1, 68, 34, 14]. The continuity equation for each phase α takes the form

$$\frac{\partial(\phi\rho_\alpha s_\alpha)}{\partial t} - \nabla\cdot(\rho_\alpha\lambda_\alpha\mathbf{K}\,(\nabla p_\alpha - \rho_\alpha g\nabla h)) + q_\alpha = 0 \tag{3.1}$$

where the subscript $\alpha = w$ denotes the water phase, the subscript $\alpha = o$ denotes the oil phase, $\mathbf{K}$ is the absolute permeability tensor, $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$ is the phase mobility, with $k_{r\alpha}$ the relative permeability to phase α and μ_α the viscosity of phase α, p_α is

the phase pressure, ρ_α is the density of phase α, g is the gravitational acceleration, h is the depth, ϕ is the porosity, s_α is the saturation of the phase α and q_α is the phase source and sink terms [1, 34]. Further, the phase saturations are constrained by $s_w + s_o = 1$, since the oil and the water jointly fill the void space [1, 68].

The phase velocities are modeled by the multiphase Darcy's law to relate the phase velocities to the phase pressures and takes the form

$$\mathbf{v}_\alpha = -\mathbf{K}\lambda_\alpha \ \nabla(p_\alpha - \rho_\alpha g h) \tag{3.2}$$

where $\mathbf{v}_\alpha$ is the phase velocity. The phase relative permeabilities $k_{r\alpha}$ and the capillary pressure ($p_{cow} = p_o - p_w$) are usually modeled as functions of the phase saturations [1]. Neglecting the capillary pressure, the compressibility effects, the gravitational effects, and assuming the density ratio to be equal to one, the continuity equations (Eq. (3.1)) can be combined with the Darcy's law (Eq. (3.2)) to derive a global pressure equation and the saturation equation for water phase [1, 68, 14]. The simplified global pressure equation takes the form

$$\nabla \cdot \mathbf{K}\lambda \ \nabla p = q \tag{3.3}$$

where $p = p_o = p_w$ is the global pressure, $\lambda = \lambda_w + \lambda_o$ is the total mobility, $q = q_w + q_o$ is the source and sink term. The saturation equation for the water phase takes the following form

$$\phi \ \frac{\partial s}{\partial t} + \mathbf{v} \cdot \nabla f_w = \frac{q_w}{\rho_w} \tag{3.4}$$

where $f_w = \lambda_w/(\lambda_w + \lambda_o)$ is a function of saturation termed as the fractional flow function for the water phase, $\mathbf{v} = -\mathbf{K}\lambda \ \nabla p$ is the total velocity vector and $s = s_w$ is the water saturation [1, 34]. In the rest of the paper, we write the water phase saturation as $s = s_w$ for simplicity. The coupled equations Eq. (3.3) and Eq. (3.4) could then be solved for the evolution of the saturation by providing the appropriate initial and boundary conditions. Equation (3.3) and Eq. (3.4) are continuous (in space and time) form of the full model.

The discrete form of the full model is obtained by dividing the problem domain

into n grid blocks and then applying the finite volume method to discretize the spatial derivatives of Eq. (3.3) and Eq. (3.4). The discretized pressure equation takes the form

$$\mathbf{A}\ \mathbf{y}_p = \mathbf{b} \tag{3.5}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, and $\mathbf{y}_p \in \mathbb{R}^n$ is the pressure vector in which each component y_{p_i} of $\mathbf{y}_p$ represent the pressure value at the ith grid block. Similarly, the spatially discretized saturation equation takes the form

$$\frac{d\mathbf{y}_s}{dt} + \mathbf{B}(\mathbf{v})\ \mathbf{f}_w(\mathbf{y}_s) = \mathbf{d} \tag{3.6}$$

where $\mathbf{B} \in \mathbb{R}^{n\times n}$, $\mathbf{d} \in \mathbb{R}^n$, $\mathbf{v}$ is the total velocity vector, and $\mathbf{y}_s \in \mathbb{R}^n$ is the saturation vector in which each component y_{s_i} of $\mathbf{y}_s$ is the saturation value at the ith grid block. We then use Euler implicit time stepping method to transform the continuous dynamical system (Eq. (3.6)) to discrete nonlinear dynamical system that takes the form

$$\mathbf{y}_s(t+1) = \mathbf{y}_s(t) - \mathbf{B}(\mathbf{v}(t))\ \mathbf{f}_w(\mathbf{y}_s(t+1))\ \Delta_t - \mathbf{d}\ \Delta_t \tag{3.7}$$

Eq. (3.5) and Eq. (3.7) are the discrete form of the full model for multi-phase flow problem under consideration. These two equations exhibit two way coupling from the dependence of the matrix $\mathbf{A}$ on the mobilities $\lambda(\mathbf{y}_s(t))$ in the pressure full model (Eq. (3.5)) and from the dependence of the matrix $\mathbf{B}$ on the velocity vector $\mathbf{v}(\mathbf{y}_p)$ in the saturation full model (Eq. (3.6)). In this study, we adopt an implicit sequential splitting method[1] to solve the full model (Eq. (3.5) and Eq. (3.7)). Figure 3.1 displays the flow chart of the numerical scheme utilized to update the pressure and saturation field over a time period T. More precisely, in this method, the saturation vector $\mathbf{y}_s(t)$ from the present time step is used to assemble the matrix $\mathbf{A}$ in Eq. (3.5) and then the pressure full model (Eq. (3.5)) is solved for the pressure vector $\mathbf{y}_p$. Following that, the velocity vector $\mathbf{v}$ (computed from the pressure vector $\mathbf{y}_p$) is used to assemble the matrix $\mathbf{B}$ in Eq. (3.6) and then the saturation full model (Eq. (3.6)) is solved implicitly in time for the saturation at the next time

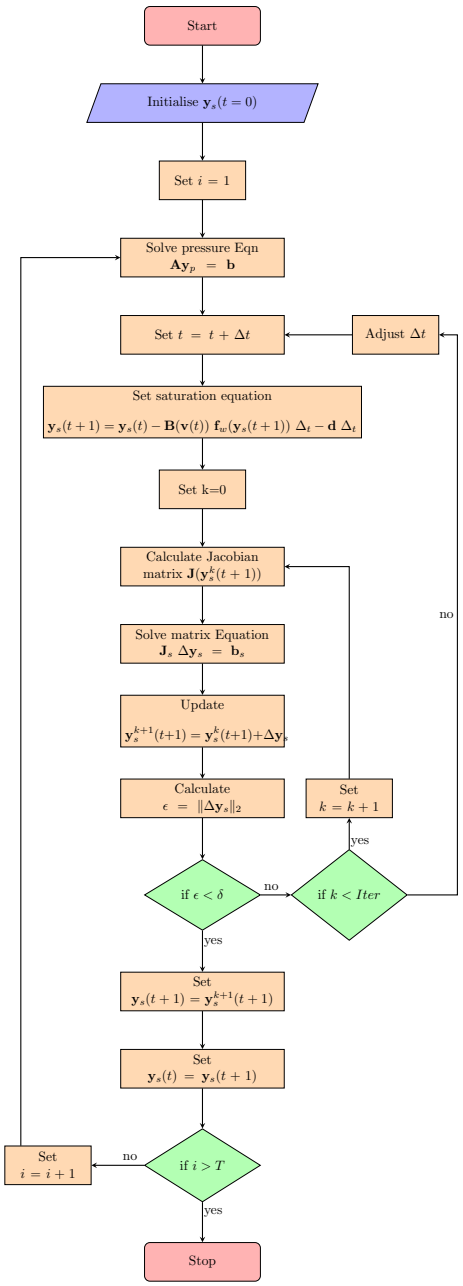


Figure 3.1: Flowchart describing the implicit sequential splitting method to solve Eq (3.5) and Eq (3.7) for a time period T.

step. In the following section, we formulate a Galerkin projection based reduced model to reduce the computational effort for multi-query tasks (e.g. uncertainty quantification) involving repeated solutions of Eq. (3.5) and Eq. (3.6), when n (the number of grid block) is large [30, 61].

3.3 Reduced Order Model Formulation

In this section, we formulate the POD-Galerkin reduced model (POD reduced model) and POD-DEIM reduced model where POD-Galerkin is combined with DEIM for handling the nonlinear terms. Both methods are introduced to reduce the computational effort associated with solving the full model (Eq. (3.5) and Eq. (3.6)).

3.3.1 POD basis

As stated in section 4.1, POD based reduced model is a projection based reduced order model in which the governing equations are projected onto an optimal low-dimensional subspace $\mathcal{U}$ spanned by a small set of r basis vectors. Galerkin projection reduced model is based on the assumption that most of the system information and characteristics can be efficiently represented by linear combinations of only a small number of basis vectors [119].

The optimal basis vectors $\{\mathbf{u}_i\}_{i=1}^{r}$ in POD are computed by singular value decomposition (SVD) of the solution snapshot matrix $\mathbf{X}$. The solution snapshot matrix $\mathbf{X}$ is obtained from a set of solution vectors of size n_s obtained by solving the full model at selected points in the input parameter space. The SVD of $\mathbf{X}$ is expressed as

$$\mathbf{X} = \mathbf{U}\ \Sigma\ \mathbf{W} \tag{3.8}$$

where, $\mathbf{X} \in \mathbb{R}^{n \times n_s}$, $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3\ \cdots\ \mathbf{u}_n] \in \mathbb{R}^{n \times n}$ is the left singular matrix and $\Sigma = \text{diag}(\sigma_1 > \sigma_2 > \sigma_3 > \cdots\ \sigma_{ns} \geq 0)$ is the diagonal matrix containing the singular values σ_i of the snapshot matrix $\mathbf{X}$ in descending order. The dominant left singular vectors $\{\mathbf{u}_i\}_{i=1}^{r}$ corresponding to the first r largest singular values represents the basis vectors to span the optimal subspace $\mathcal{U}$ of POD based reduced model. Thus, the first step in deriving the POD based reduced model is to express the state vector $\mathbf{y}$ of the full-order model by a linear combination of r basis vectors as following

$$\mathbf{y} \approx \mathbf{U}^r\ \tilde{\mathbf{y}} \tag{3.9}$$

where $\tilde{\mathbf{y}} \in \mathbb{R}^r$ is the reduced state vector representation of full dimensional state

vector $\mathbf{y}$, and $\mathbf{U}^r = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_r] \in \mathbb{R}^{n \times r}$ is the matrix that contains r orthonormal basis vectors in its columns.

By following this step, for example, the optimal basis vectors for the saturation state vector $\mathbf{y}_s$ are obtained from the SVD of the saturation snapshot matrix $\mathbf{X}_s = \big((\mathbf{y}_{s_1} \ \cdots \ \mathbf{y}_{s_T})^1 \ \cdots \ (\mathbf{y}_{s_1} \ \cdots \ \mathbf{y}_{s_T})^L\big)$, where T is the number of time steps and L is the number of samples of input parameter used to build the snapshot matrix. The SVD of $\mathbf{X}_s$ is expressed as

$$\mathbf{X}_s = \mathbf{U}_s \ \Sigma_s \ \mathbf{W}_s \tag{3.10}$$

where $\mathbf{U}_s \in \mathbb{R}^{n \times n}$ is the left singular matrix, Σ_s is the diagonal matrix containing the singular values of the snapshot matrix $\mathbf{X}_s$ in descending order. The saturation state vector $\mathbf{y}_s$ is optimally expressed as

$$\mathbf{y}_s \approx \mathbf{U}_s^r \ \tilde{\mathbf{y}}_s \tag{3.11}$$

where $\tilde{\mathbf{y}}_s \in \mathbb{R}^r$ is the reduced state vector representation of $\mathbf{y}_s$, $\mathbf{U}_s^r \in \mathcal{R}^{n \times r}$ is the matrix that contains r orthonormal basis vectors in its columns. Similarly, we can represent the pressure state vector $\mathbf{y}_p$ from its reduced state vector representation $\tilde{\mathbf{y}}_p$ using optimal basis matrix $\mathbf{U}_p$ obtained from the SVD of the pressure snapshot matrix $\mathbf{X}_p$.

3.3.2 Least–squares approximation

The capacity of a set of basis functions to represent a new solution vector could be tested using least–squares fitting [46, 131]. For example, the least–squares solution for approximating a saturation state vector $\mathbf{y}_s^* \in \mathbb{R}^n$ is defined as

$$\mathbf{y}_s^* \approx \mathbf{U}_s^r \ \tilde{\mathbf{y}}_s = \mathbf{U}_s^r \ (\mathbf{U}_s^{r\top} \ \mathbf{y}_s) \tag{3.12}$$

The associated error termed as least–squares errors in approximating $\mathbf{y}_s$ by $\mathbf{y}_s^*$ using only r basis vectors is given by

$$\varepsilon_s = \|\mathbf{y}_s - \mathbf{y}_s^*\|_2 \tag{3.13}$$

where $\|\cdot\|_2$ denotes Euclidean norm which comes under a class of vector norm called p-norm [131, 46] as defined in Eq. (2.18) chapter 2. The least–squares error ε_s (Eq. (3.13)) is equivalent to the omitted energy $\Omega_s = \sum_{i=r+1}^{n} \sigma_{s_i}$ [96, 17]. In practice, r is commonly chosen as the smallest integer such that the relative omitted energy ν is less than a preset value (e.g. 0.01), where the omitted energy is defined by the following equation

$$\nu = 1 - \frac{\sum_{i=r+1}^{n} \sigma_{s_i}}{\sum_{i=1}^{n} \sigma_{s_i}} \tag{3.14}$$

Similar expressions mentioned in Eq. (3.12), Eq. (3.13), and Eq. (3.14) can be obtained for the pressure state vector as well. We note that least–squares errors are not necessarily equivalent to the omitted energy for state vectors not included in the snapshot matrix or for the state vector solved at a new point in the input parameter space as these new vectors might not fall within the span of the snapshot matrix [53, 96]. The least–squares solution is the best approximation of the state variables in the sense that, for the chosen low dimensional subspace $\mathcal{U}$, no other low dimensional approximation can represent the given snapshot set with a lower least–squares error [89, 125, 17]. In this chapter, we use the best approximation of the state variables to assess the quality of the approximation obtained from different reduced order models in the numerical examples presented later in section 3.5.

3.3.3 POD-Galerkin

Once the POD basis vectors are obtained, the reduced representation of the pressure vector $\mathbf{y}_p$ is substituted into the pressure full model (Eq. (3.5)), followed by Galerkin projection of the pressure equation into the subspace spanned by $\mathbf{U}_p^r$. The resulting POD based reduced model for the pressure equation then takes the following form

$$\tilde{\mathbf{A}}\ \tilde{\mathbf{y}}_p = \tilde{\mathbf{b}} \tag{3.15}$$

where $\tilde{\mathbf{A}} = \mathbf{U}_p^{r\top}\ \mathbf{A}\ \mathbf{U}_p^r \in \mathbb{R}^{r\times r}$ and $\tilde{\mathbf{b}} = \mathbf{U}_p^{r\top}\ \mathbf{b} \in \mathbb{R}^r$. Similarly, POD based reduced model for the saturation equation (Eq. (3.6)) takes the form

$$\frac{d\tilde{\mathbf{y}}_s}{dt} + \mathbf{U}_s^{r\top}\ \mathbf{B}(\mathbf{v})\ \mathbf{f}_w(\mathbf{U}_s^r\ \tilde{\mathbf{y}}_s) = \tilde{\mathbf{d}}, \tag{3.16}$$

where $\tilde{\mathbf{d}} = \mathbf{U}_s^{r\top}\ \mathbf{d}$ and $\tilde{\mathbf{d}} \in \mathbb{R}^r$.

The POD based reduced model formulated by Eq. (3.15) and Eq. (3.16) is of the reduced dimension r. However, the nonlinear function $\mathbf{f}_w$ in Eq. (3.16) is still of the order of full dimension n. Moreover, the reduced Jacobian matrix $\tilde{\mathbf{J}} = \tilde{\mathbf{I}} - \mathbf{U}_s^{r\top}\mathbf{B}\ \mathbf{J}_f(\mathbf{f}_w(\mathbf{U}_s^r\ \tilde{\mathbf{y}}_s))\mathbf{U}_s^r \in \mathbb{R}^{r\times r}$ needed for Newton like iterations to solve this nonlinear equation is also of order n [30] as it relies on evaluating the full order nonlinear function $\mathbf{f}_w$. Therefore, for problems with general nonlinear functions involved in POD based reduced model, the computational cost of solving the reduced system is still a function of the full system dimension n.

3.3.4 DEIM

Discrete Empirical Interpolation Method (DEIM) was introduced in [30] to approximate the nonlinear terms in POD based reduced model using a limited number of points that are independent of the full system dimension n. Similar to POD, the first step of DEIM is to approximate the nonlinear function $\mathbf{f}_w$ in Eq. (3.16) using a separate set of basis vectors $\mathbf{V}^m = [\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\ \ldots\ \mathbf{v}_m]$ as

$$\mathbf{f}_w = \mathbf{V}^m\ \tilde{\mathbf{f}} \tag{3.17}$$

where $\tilde{\mathbf{f}}$ is the coefficient of expansion of the nonlinear function $\mathbf{f}_w$ in the reduced subspace spanned by $\{\mathbf{v}_i\}_{i=1}^m$, $\mathbf{V}^m \in \mathbb{R}^{n\times m}$ is the matrix containing the first m columns of the left singular matrix $\mathbf{V} \in \mathbb{R}^{n\times n}$ obtained from the SVD of the the snapshot matrix $\mathbf{X}_f$ of the nonlinear function $\mathbf{f}_w$. We note that no additional computational costs are associated with collecting the snapshot matrix of the nonlinear terms $\mathbf{X}_f$ as it is already evaluated during the computation of the state snapshot

vectors. The nonlinear term in Eq. (3.16) can then be expressed as

$$\mathbf{U}_s^{r\top}\ \mathbf{B}\ \mathbf{f}_w = (\mathbf{U}_s^{r\top}\ \mathbf{B}\ \mathbf{V}^m)\ \tilde{\mathbf{f}} = (\mathbf{U}_s^{r\top}\ \mathbf{B}\ \mathbf{V}^m)\ \cdot\ (\mathbf{V}^{m\top}\ \mathbf{f}_w) \tag{3.18}$$

The matrix factor $(\mathbf{U}_s^{r\top}\ \mathbf{B}\ \mathbf{V}^m) \in \mathbb{R}^{r\times m}$ in Eq. (3.18) is precomputed before solving Eq. (3.16). The overdetermined system $\tilde{\mathbf{f}} = \mathbf{V}^{m\top}\ \mathbf{f}_w$ is approximated using the DEIM algorithm introduced in [30] by first computing a matrix $\mathbf{P} \in \mathbb{R}^{n\times m}$ that selects m rows of the matrix $\mathbf{V}^m$ to obtain $\tilde{\mathbf{f}}$ as following

$$\mathbf{P}^\top\ \mathbf{f}_w = \mathbf{P}^\top\ \mathbf{V}^m\ \tilde{\mathbf{f}} \Longleftrightarrow \tilde{\mathbf{f}} = (\mathbf{P}^\top\ \mathbf{V}^m)^{-1}\ \mathbf{P}^\top\ \mathbf{f}_w \tag{3.19}$$

Using this expression of $\tilde{\mathbf{f}}$ to approximate the nonlinear function in Eq. (3.18), we obtain a nonlinear term that is independent of n that takes the form

$$\mathbf{U}_s^{r\top}\ \mathbf{B}\ \mathbf{f}_w \approx \mathbf{D}\ \mathbf{f}_w(\mathbf{P}^\top\ \mathbf{U}_s^r\ \tilde{\mathbf{y}}_s) \tag{3.20}$$

where the matrix $\mathbf{D} = \mathbf{U}_s^{r\top}\ \mathbf{B}\ \mathbf{V}^m\ (\mathbf{P}^\top\ \mathbf{V}^m)^{-1} \in \mathbb{R}^{r\times m}$ termed as the DEIM matrix. Similarly, the Jacobian of the nonlinear term in Eq. (3.16) is approximated using DEIM as following

$$\tilde{\mathbf{J}} = \tilde{\mathbf{I}} - (\mathbf{U}_s^{r\top}\mathbf{B}\mathbf{V}^m(\mathbf{P}^\top\ \mathbf{V}^m)^{-1})\ \hat{\mathbf{J}}_f(\mathbf{f}_w(\mathbf{P}^\top\ \mathbf{U}_s^r\ \tilde{\mathbf{y}}_s))\ (\mathbf{P}^\top\mathbf{U}_s^r) \tag{3.21}$$

where $\hat{\mathbf{J}}_f(\mathbf{f}_w(\mathbf{P}^\top\ \mathbf{U}_s^r\ \tilde{\mathbf{y}}_s)) \in \mathbb{R}^{m\times m}$ is the Jacobian matrix computed using the m components of $\mathbf{f}_w$ evaluated by the DEIM algorithm [30, 119, 104]. Finally, the POD-DEIM based reduced model takes the form

$$\frac{d\tilde{\mathbf{y}}_s}{dt} + \mathbf{D}\ \mathbf{f}_w(\mathbf{P}^\top\ \mathbf{U}_s^r\ \tilde{\mathbf{y}}_s) = \tilde{\mathbf{d}} \tag{3.22}$$

We note that POD-DEIM formulation is independent of the full model dimension n and that the DEIM procedure exploits the structure of the nonlinear function $\mathbf{f}_w$ as component-wise operation at $\mathbf{U}_s^r\ \tilde{\mathbf{y}}_s$ [30].

3.4 Deep Residual RNN

POD-DEIM reduced order models, as introduced in the last chapter, could be used to perform parametric UQ tasks. However, the POD-DEIM formulation is nonlinear and relies on using Newton method at each time step to solve the resulting system of nonlinear equations. The computational efficiency of the Newton iteration depends on the method employed to assemble the Jacobian matrix and more importantly on the conditioning of the reduced Jacobian matrix. It also depends on the method used to solve the resulting linear system at each iteration of the Newton step and generally, it takes $\mathcal{O}(r^3)$ operations for each saturation update [104, 18]. Moreover, previous studies [68, 67] pointed to the loss of stability of POD-Galerkin reduced model in several cases and it was attributed to ill-conditioning and poor spectral properties of the reduced Jacobian matrix.

In this paper, we build on the recently introduced DR-RNN [104] and formulate an accurate POD-DEIM reduced order models. DR-RNN is a deep RNN architecture [104], constructed by stacking K physics aware network layers. DR-RNN could be applied to any nonlinear dynamical system of the form

$$\frac{d\mathbf{y}}{dt} = \mathbf{A}\ \mathbf{y} + \mathbf{F}(\mathbf{y}) \tag{3.23}$$

where $\mathbf{y}(\mathbf{a}, t) \in \mathbb{R}^n$ is the state variable at time t, $\mathbf{a} \in \mathbb{R}^d$ is a system parameter vector, the matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the linear part of the dynamical system and the vector $\mathbf{F}(\mathbf{y}) \in \mathbb{R}^n$ is the nonlinear term [104]. The state variable $\mathbf{y}(t)$ at different time steps is obtained by solving the nonlinear residual equation defined as

$$\mathbf{r}_{t+1} = \mathbf{y}_{t+1} - \mathbf{y}_t - \Delta t\ \mathbf{A}\ \mathbf{y}_{t+1} - \Delta t\ \mathbf{F}(\mathbf{y}_{t+1}) \tag{3.24}$$

where $\mathbf{r}(t)$ is termed as the residual vector at time step t and $\mathbf{y}(t+1)$ is the approximate solution of Eq. (3.23) at time step $t+1$ obtained by using implicit Euler time integration method. DR-RNN [104] approximates the solution of Eq. (3.23) using

the following iterative update equations

$$\begin{aligned} \mathbf{y}_{t+1}^{(k)} &= \mathbf{y}_{t+1}^{(k-1)} - \mathbf{w} \ \circ \ \phi_h(\mathbf{U}\ \mathbf{r}_{t+1}^{(k)}) \qquad && \text{for } k = 1, \\ \mathbf{y}_{t+1}^{(k)} &= \mathbf{y}_{t+1}^{(k-1)} - \frac{\eta_k}{\sqrt{G_k + \epsilon}}\ \mathbf{r}_{t+1}^{(k)} && \text{for } k > 1, \end{aligned} \tag{3.25}$$

where $\mathbf{U}, \mathbf{w}, \eta_k$ are the training parameters of DR-RNN, ϕ_h is the tanh activation function, $\circ$ is an element-wise multiplication operator, $\mathbf{r}_{t+1}^{(k)}$ is the residual in layer k obtained by substituting $\mathbf{y}_{t+1} = \mathbf{y}_{t+1}^{(k-1)}$ into Eq. (3.24) and G_k is an exponentially decaying squared norm of the residual defined by

$$G_k = \gamma\ \|\mathbf{r}_{t+1}^{(k)}\|_2^2 + \zeta\ G_{k-1} \tag{3.26}$$

where γ, ζ are fraction factors and ϵ is a smoothing term to avoid divisions by zero [104]. In this formulation, we set $\mathbf{y}_{t+1}^{(k=0)} = \mathbf{y}_t$. The architecture of DR-RNN is inspired by the rmsprop algorithm [130] which is a variant of the steepest descent method. The DR-RNN output at each time step is defined as

$$\mathbf{y}_{t+1}^{(\text{RNN})} = \mathbf{y}_{t+1}^{K} \tag{3.27}$$

The formulation of DR-RNN is explicit in time and has a fixed number of iterations K per time step. However, the dimension of the DR-RNN system depends on the dimension of the residual. For example, DR-RNN (Eq. (3.25)) can be derived from the POD-DEIM reduced model residual ($\tilde{\mathbf{r}}_{t+1} = -\tilde{\mathbf{y}}_{s_{t+1}} + \tilde{\mathbf{y}}_{s_t} + \mathbf{D}\ \mathbf{f}_w(\mathbf{P}^\top\ \mathbf{U}_s^r\ \tilde{\mathbf{y}}_{s_{t+1}}) + \tilde{\mathbf{d}}$). In such setting, the DR-RNN dynamics has a fixed computational budget of $\mathcal{O}(r^2)$ for each time step. In addition, DR-RNN has the prospect of employing large time step violating the numerical stability constraint [104]. Furthermore, DR-RNN does not rely on the reduced Jacobian matrix to approximate the solution of POD-DEIM reduced model.

The DR-RNN parameters $\boldsymbol{\theta} = \{\mathbf{U},\ \mathbf{w},\ \eta_k\}$ are fitted by minimizing the mean

square error (mse) defined by

$$\mathbf{J}_{\text{MSE}}(\boldsymbol{\theta}) = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{t=1}^{T}(\mathbf{y}_t - \mathbf{y}_t^{(\text{RNN})})^2, \tag{3.28}$$

where $\mathbf{J}_{\text{MSE}}$ (mse) is the average distance between the reference solution $\mathbf{y}_t$ and the RNN output $\mathbf{y}_t^{\text{RNN}}$ across a number of samples L with time-dependent observations ($t = 1 \cdots T$ and $\ell = 1 \cdots L$) [104, 107]. The set of parameters $\boldsymbol{\theta}$ is commonly estimated by a technique called Backpropagation Through Time (BPTT) [138, 122, 108, 100], which backpropagates the gradient of the loss function $\mathbf{J}_{\text{MSE}}$ with respect to $\boldsymbol{\theta}$ in time over the length of the simulation.

3.5 Numerical Experiments

In this section, we evaluate the performance of the reduced order models based on DR-RNN against the standard implementation of POD-Galerkin reduced model. Specifically, we develop two POD-Galerkin based reduced model using DR-RNN architecture namely, DR-RNN$^{\text{p}}$ (DR-RNN combined with POD-Galerkin) and DR-RNN$^{\text{pd}}$ (DR-RNN combined with POD-Galerkin and DEIM). The numerical evaluations are performed using two uncertainty quantification tasks involving subsurface flow models. We did not include standard POD-DEIM reduced model implementation as we expect that the standard POD reduced model results to be far superior [30, 104, 30].

The outline of this section is as follows: In subsection 3.5.1, we present the description of the flow problem, followed by a brief description of the finite-volume approach employed for obtaining the full-order model solution. Following that, in subsection 3.5.2, we outline the specific details to formulate POD reduced model. Then, we list the settings adopted to model the DR-RNN ROMs (i.e. number of layers, optimization settings, etc) in the subsection 3.5.3. In subsection 3.5.4, we provide a set of error metrics utilized to evaluate the performance of the different ROMs. In subsection 3.5.5, we present the numerical results for the quarter five spot model followed by results for the uniform flow model in the subsection 3.5.6.

3.5.1 Full-order model setup

We consider a two-phase (oil and water) porous media flow problem over the two-dimensional domain $[0\ 1] \times [0\ 1]$. The equations governing the two-phase flow are the pressure equation (Eq. (3.3)) and the saturation equation (Eq. (3.4)). The relative permeability is defined as a function of saturation using Corey's model $k_{rw}(s) = s^{*2}$, $k_{ro} = (1-s^*)^2$, where $s^* = (s-s_{wc})/(1-s_{or}-s_{wc})$, s_{wc} is the irreducible water saturation and s_{or} is the residual oil saturation [1]. We set $s_{or} = 0.2$ and $s_{wc} = 0.2$. We set the initial water saturation over the domain to the irreducible water saturation $s_{wc} = 0.2$. The water to oil viscosity ratio is set to 0.7. The porosity is assumed to be a constant value of 0.2 over the entire problem domain. The uncertain permeability field is modeled as a log-normal distribution function with zero mean and exponential covariance kernel of the form

$$\mathbb{C}ov = \sigma_k \ \exp\left[-\frac{|x_1 - x_2|}{L_k}\right] \tag{3.29}$$

where σ_k is the variance, L_k is the correlation length, x_1 and x_2 are the points in the grid domain. In all test cases, we set σ_k to 1 and the correlation length L_k to 0.1. Figure 3.2 shows several realizations of the log-permeability values. For solving the

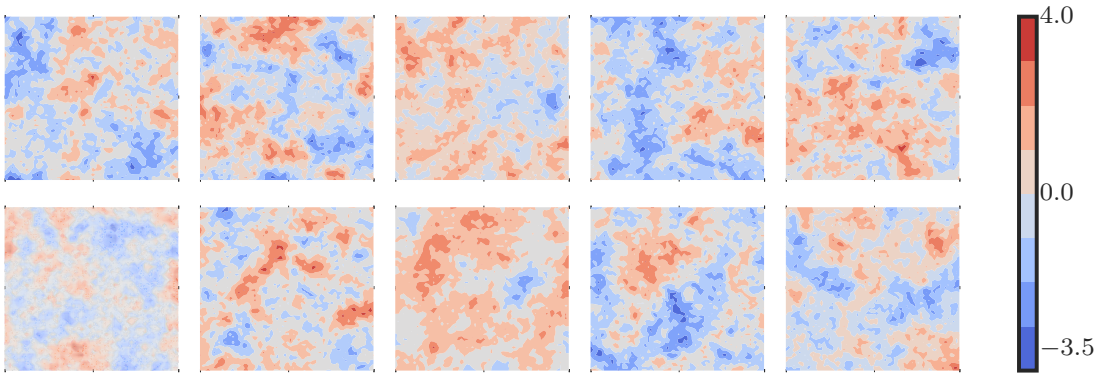


Figure 3.2: Plots of log values of random permeability field modeled by log-normal probability distribution.

full-order model, the problem domain is discretized using a uniform grid of 64×64 blocks. The pressure equation is discretized using simple finite volume method (aka. Two Point Flux Approximation) [1] and an upwind finite-volume scheme is

used to discretized the saturation equation. For the time discretization, an implicit backward Euler method combined with Newton-Raphson iterative method is used to solve the resulting system of nonlinear equations. We set the time step size to 0.015 and the total number of time steps is set to 160. We note that, the time is measured in a non-dimensional quantity called pore volumes injected (PVI). PVI defines the net volume of water injected as a fraction of the total pore volume. As the pressure changes at much slower rate than the saturation, the pressure equation (and hence the velocity) is solved at every 8th saturation time step. For reference solutions, this system of equations is solved for 2000 random permeability realizations to estimate an ensemble based statistics using Monte-Carlo method [78]. We implemented the pressure and saturation equation solver in python programing language and the code is attached in the appendix. Please also refer Figure 3.1 for the pictorial representation of the pseudo code.

3.5.2 POD-Galerkin based reduced model setup

The first step in formulating POD reduced model is to compute the optimal POD basis matrices $\mathbf{U}_p^r$ and $\mathbf{U}_s^r$. In order to obtain these basis matrices, we initially preformed a realization clustering algorithm to enforce the diversity of the collected snapshots and clustered the 2000 random permeability realizations into 45 clusters [61]. Then, we randomly selected a single permeability realization from each cluster (total 45 random samples of the permeability field). The full system is then solved for each of the 45 realizations and the solution vectors are collected to build the snapshot matrices (pressure, saturation, nonlinear function). Finally, we compute the POD basis matrices from the SVD of the collected snapshot matrices.

Following that, the obtained basis vectors are used to build POD reduced model (as detailed in the section 3.3). We then employ the same sequential implicit technique settings adopted for obtaining the full model solutions to solve the resultant POD reduced model. For numerical evaluations, we solve the POD reduced model for the same 2000 permeability realizations to estimate an ensemble based statistics in the engineering quantities of interest.

3.5.3 DR-RNN setup

In all the numerical test cases, we utilize DR-RNN with six layers ($K = 6$ in Eq. (3.25)). We evaluate DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ for different number of POD basis, however, we fix the number of DEIM basis to 35. The PyTorch framework [111], a deep learning python package using Torch library as a backend, is used to implement the DR-RNN. Further, we optimize the DR-RNN model parameters using rmsprop algorithm [130, 111] as implemented in PyTorch, where we set the weighted average parameter to 0.9 and the learning rate to 0.001. The weight matrix $\mathbf{U}$ in Eq. (3.25) is initialized randomly from the uniform distribution function $\mathtt{U}[0.01, 0.02]$. The vector training parameter $\mathbf{w}$ in Eq. (3.25) is initialized randomly from the uniform distribution function $\mathtt{U}[0.1, 0.5]$. The scalar training parameters η_k in Eq. (3.25) are initialized randomly from the uniform distribution $\mathtt{U}[0.1, 0.4]$. We set the hyperparameters ζ and γ in Eq. (3.26) to 0.9 and 0.1, respectively. The formulated DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ are trained to approximate the reduced state vector representation obtained from least–squares fits. Specifically, we collect a set of best reduced state vector representation $\tilde{\mathbf{y}}_s^*$ of the saturation state vector using $\tilde{\mathbf{y}}_s^* = \mathbf{U}_s^{r\top}\ \mathbf{y}_s$. The collected set of reduced state vectors is then used to train the parameters of the DR-RNN by minimizing the loss function defined in Eq. (3.28).

3.5.4 Evaluation metrics

We evaluate the performance of DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ using two time specific error metrics defined by

$$\begin{aligned} L_{2_{l,t}} &= \| \left(\mathbf{y}_t - \mathbf{y}_t^{(\text{RM})}\right)^l \|_2 \\ L_{\infty_{l,t}} &= \| \left(\mathbf{y}_t - \mathbf{y}_t^{(\text{RM})}\right)^l \|_\infty \end{aligned} \tag{3.30}$$

where l is the realization index, and $\mathbf{y}_t^{(\text{RM})}$ is computed from the reduced model, $\|\cdot\|_2$ denotes Euclidean norm and $\|\cdot\|_\infty$ denotes ∞ norm which come under a class of vector norm called p-norm [131, 46] as defined in Eq.(2.18) chapter 2. Additionally,

we utilize two relative error metrics defined as

$$
\begin{aligned}
L_2^{\text{rel}} &= \frac{1}{L \times T} \sum_{\ell=1}^{L} \sum_{t=1}^{T} \left\| \left(\frac{\mathbf{y}_t - \mathbf{y}_t^{(\text{RM})}}{\mathbf{y}_t} \right)^l \right\|_2 \\
L_{2,\max}^{\text{rel}} &= \max_{l,t=1 \text{ to } L,T} \left\| \left(\frac{\mathbf{y}_t - \mathbf{y}_t^{(\text{RM})}}{\mathbf{y}_t} \right)^l \right\|_2
\end{aligned}
\tag{3.31}
$$

where all the time snapshots of saturation vectors in all realizations are used.

3.5.5 Numerical test case 1

In this test case, water is injected at the lower left corner $(0, 0)$ of the domain and a mixture of oil and water is produced at the top right corner of the domain $(1, 1)$. We set the injection rate $q = 0.05$ at $(0, 0)$ and $q = -0.05$ at $(1, 1)$ as defined in Eq. (3.4). We impose a no flow boundary condition in all the four sides of the domain. We fix the number of pressure POD basis to 5 and obtain all the ROMs for a set of different number of saturation POD basis functions ($r = 10, 20$). The configuration of the problem domain is shown in top left panel of Figure 3.3, where the blue spot in the lower left corner (0,0) corresponds to the injector well and the blue spot in the upper right corner (1,1) corresponds to the production well. Figure 3.3 shows the singular values of the pressure snapshot matrix $\mathbf{X}_p$ in the top right panel, the saturation snapshot matrix $\mathbf{X}_s$ in the bottom left panel, and the nonlinear function snapshot matrix $\mathbf{X}_f$ in the bottom right panel.

The mean water saturation plots over the simulation time are shown in Figure 3.4, where the results in the top row corresponds to using 10 POD basis and the results in the bottom row corresponds to using 20 POD basis. The sub-plots in Figure 3.4 are arranged from left to right following the numbering of the spatial points shown in Figure 3.3. From these results, it is clear that DR-DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ results are very close to the least–square solutions (LS fit). In Figure 3.4, POD-Galerkin reduced model yields extremely inaccurate and unstable results. We attribute the small errors in DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ results to the insufficient number of POD basis vectors and we note that the error magnitude is equivalent to the optimal values obtained by least-squares projection.

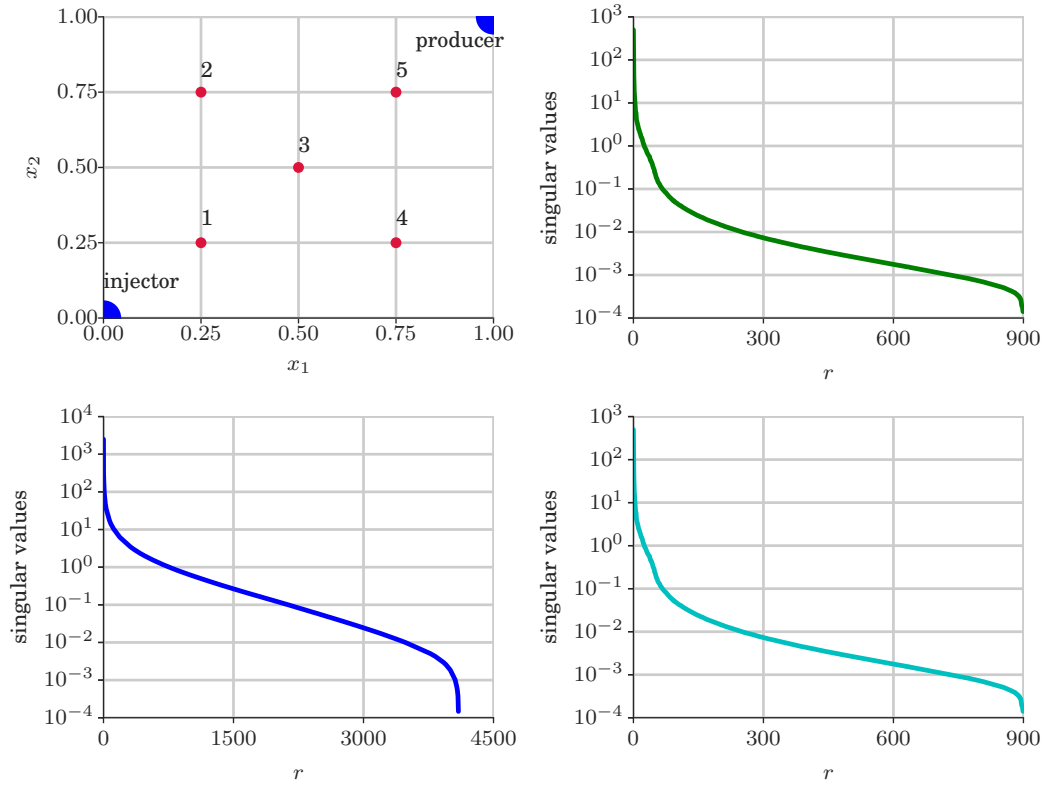


Figure 3.3: Top Left: Computational porous media domain in test case 1. The blue dot in the lower left corresponds to the injector well and the blue dot in the upper right corner corresponds to the production well. The red dots represented in numbers from 1 to 5 corresponds to the locations where the PDF and the water saturation are investigated. Top Right: Singular values of the pressure snapshot matrix $\mathbf{X}_p$. Bottom Left: Singular values of the saturation snapshot matrix $\mathbf{X}_s$. Bottom Right: Singular values of the nonlinear function snapshot matrix $\mathbf{X}_f$

Figures 3.5, 3.6, and 3.7 show the results for the first (mean) and second (standard deviation) moments of the saturation field at time = 0.3 PVI obtained from the full model and from the various ROMs. In these figures (3.5, 3.6, and 3.7), results for 10 POD basis are shown in the top row and results for 20 POD basis are shown in the bottom row. As shown in Figure 3.5, the mean saturation obtained from DR-RNN ROMs are almost indistinguishable from the reference results. However, the mean saturation field obtained from POD reduced model (left panels of Figure 3.7) deviates significantly from the reference mean saturation.

In Figure 3.6, we observe small discrepancy of standard deviation results obtained the DR-RNN ROMs in comparison to the full model results especially near the location of the mean water saturation front. Figure 3.7 (right panels) shows the standard deviation results obtained by POD reduced model which show significant

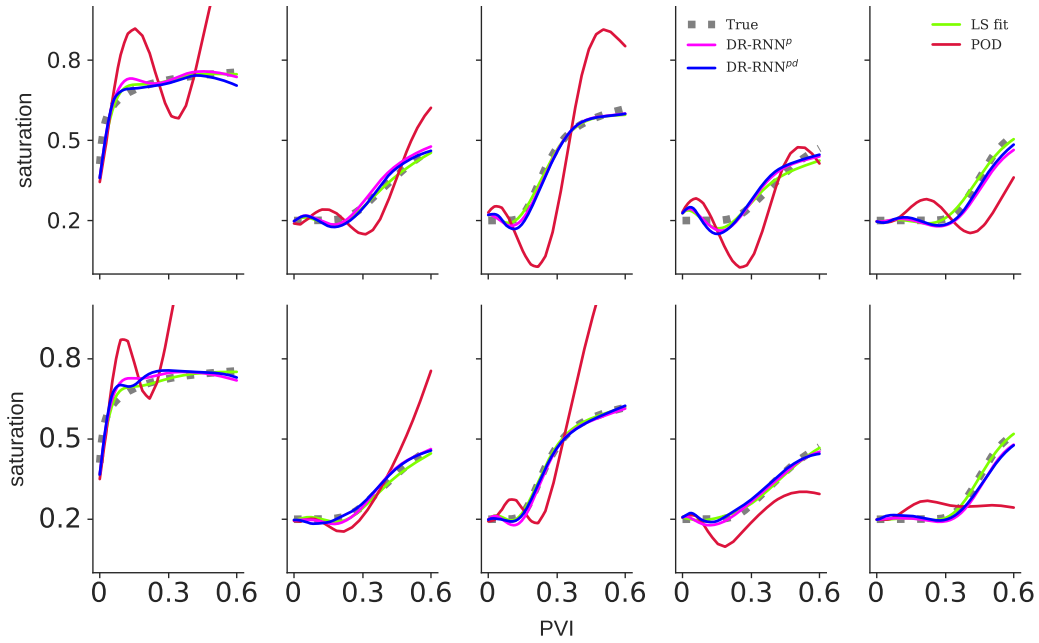


Figure 3.4: Time plots of mean water saturation obtained from all the ROMs and the full-order model for test case 1. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20. The plots in each row are arranged as per the numerical notation of the spatial points plotted in Figure 3.3 (top left panel).

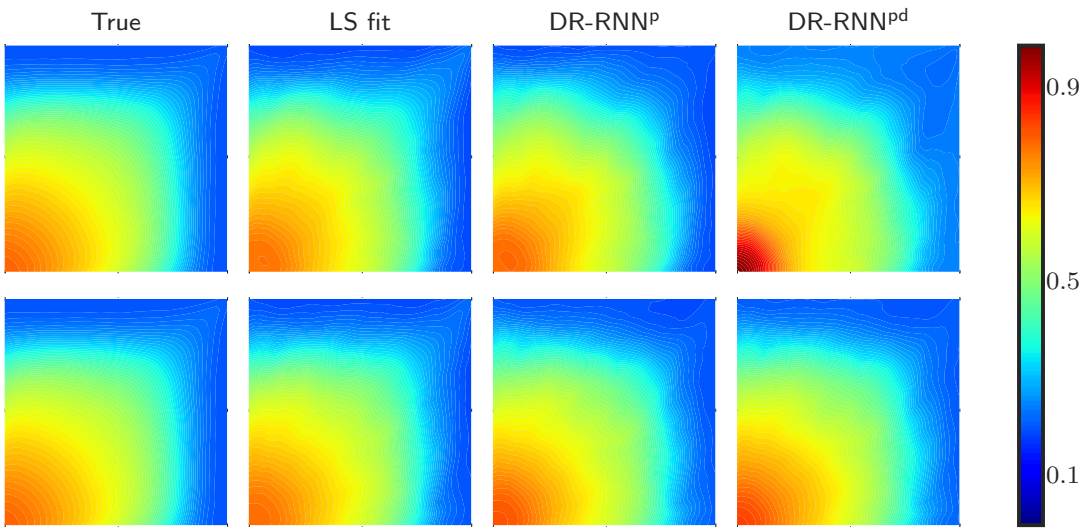


Figure 3.5: Comparison of mean water saturation field at time = 0.3 PVI for test case 1. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20.

inaccuracies that could be indicative to instabilities of the obtained solutions. We note that the white spots in Figure 3.7 correspond to out of limits shown in colorbar.

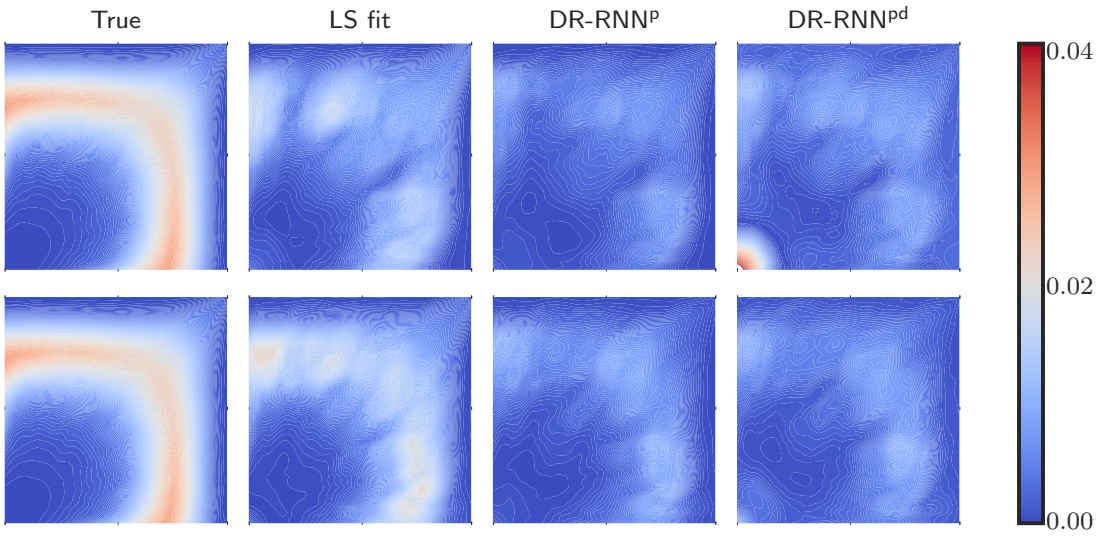


Figure 3.6: Comparison of standard deviation of the water saturation field at time = 0.3 PVI for test case 1. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20.

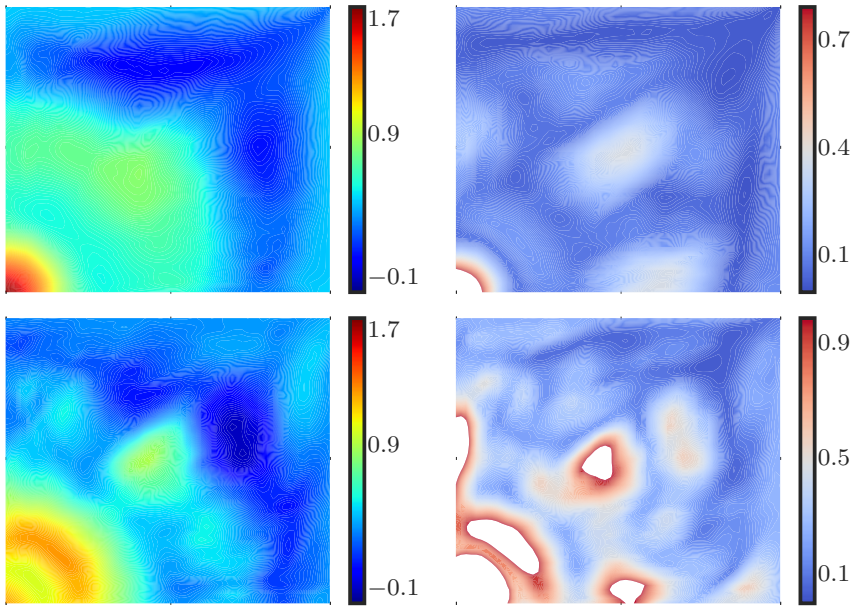


Figure 3.7: Plot of saturation mean and standard deviation of the water saturation field at time = 0.3 PVI obtained from the POD reduced model for test case 1. Left: saturation mean. Right: standard deviation. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20.

Figure 3.8 compares the saturation PDF estimated from the ensemble of numerical solutions (ROMs and the full model). Figure 3.8 settings are similar to the one adopted in Figure 3.4. In Figure 3.8, we can see that all the plots obtained from DR-DR-RNN^{p} and DR-RNN^{pd} are indistinguishable from the plots obtained

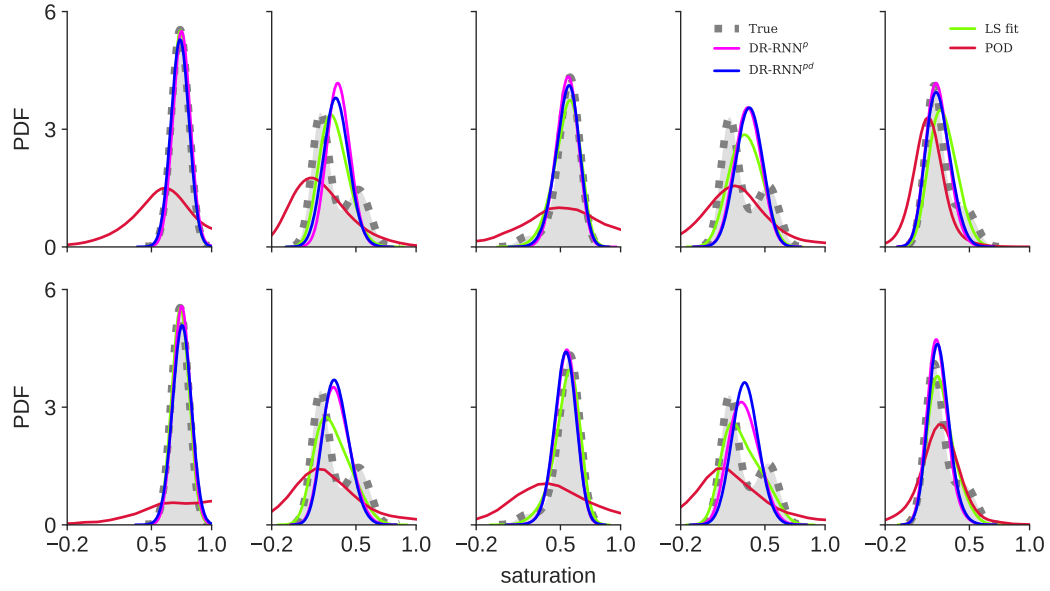


Figure 3.8: Comparison of kernel density estimated probability density function (PDF) at time = 0.3 PVI for test case 1. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20. The plots in each row are arranged as per the numerical notation of the spatial points plotted in Figure 3.3 (top left panel).

from the LS fit (the best approximation). Further, we observe that the saturation PDF obtained from DR-DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ follow nearly the same trend of saturation PDF obtained from the full model when the reference distribution is unimodal. However, we observe some discrepancy when the distributions are multimodal. Please note that similar discrepancy is also observed in the PDF obtained from LS fit. Hence, we postulate that these discrepancies are attributed to the limited number of POD basis vectors utilized. In Figure 3.8, POD reduced model yields very inaccurate approximation of the saturation PDF irrespective of the number of POD basis.

Figures 3.9 and 3.10 displays samples of $\log(L_{2_{l,t}})$ and $\log(L_{\infty_{l,t}})$ errors at time 0.3 PVI obtained from all the ROMs. All the ROMs use 10 POD basis to display the errors in Figure 3.9 and likewise 20 POD basis to display the errors in Figure 3.10. From these figures, we can see that the POD reduced model approximation errors are at least an order of magnitude more than the least–squares solution errors (Eq. (3.12)), whereas the errors obtained from DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ are nearly indistinguishable from the least–squares projection errors.

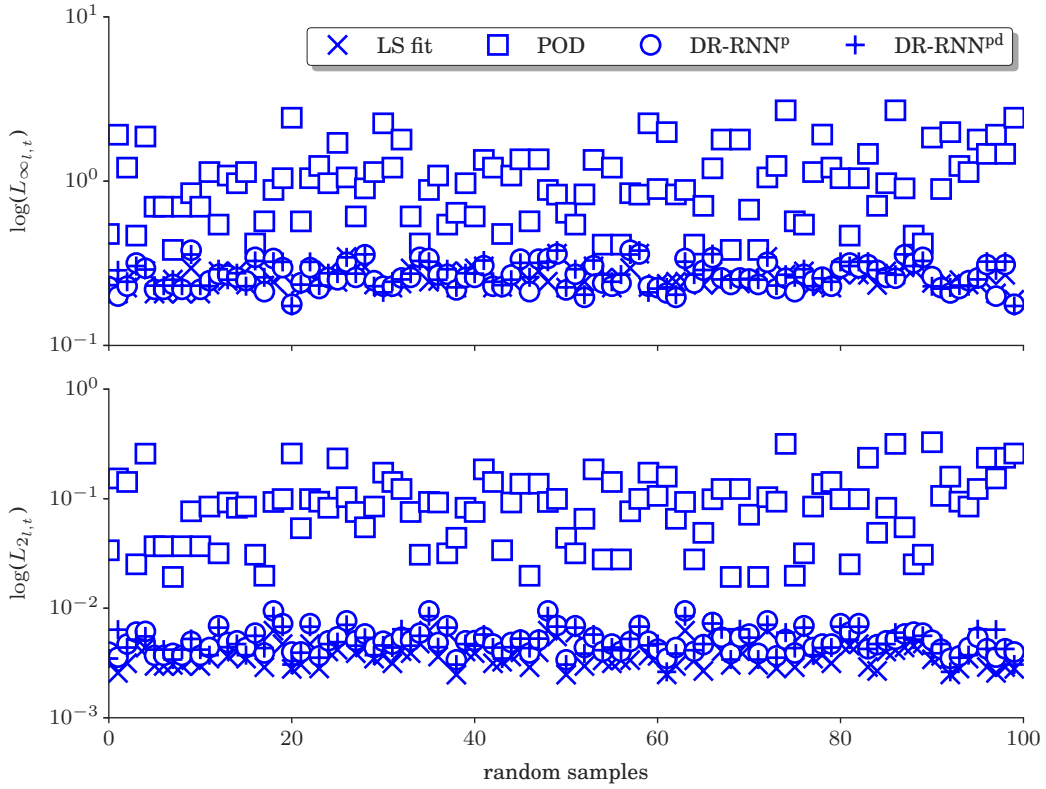


Figure 3.9: Comparison of $\log(L_{2_{l,t}})$ and $\log(L_{\infty_{l,t}})$ error estimators (Eq. (3.30)) at time = 0.3 PVI for test case 1. The number of POD basis used = 10.

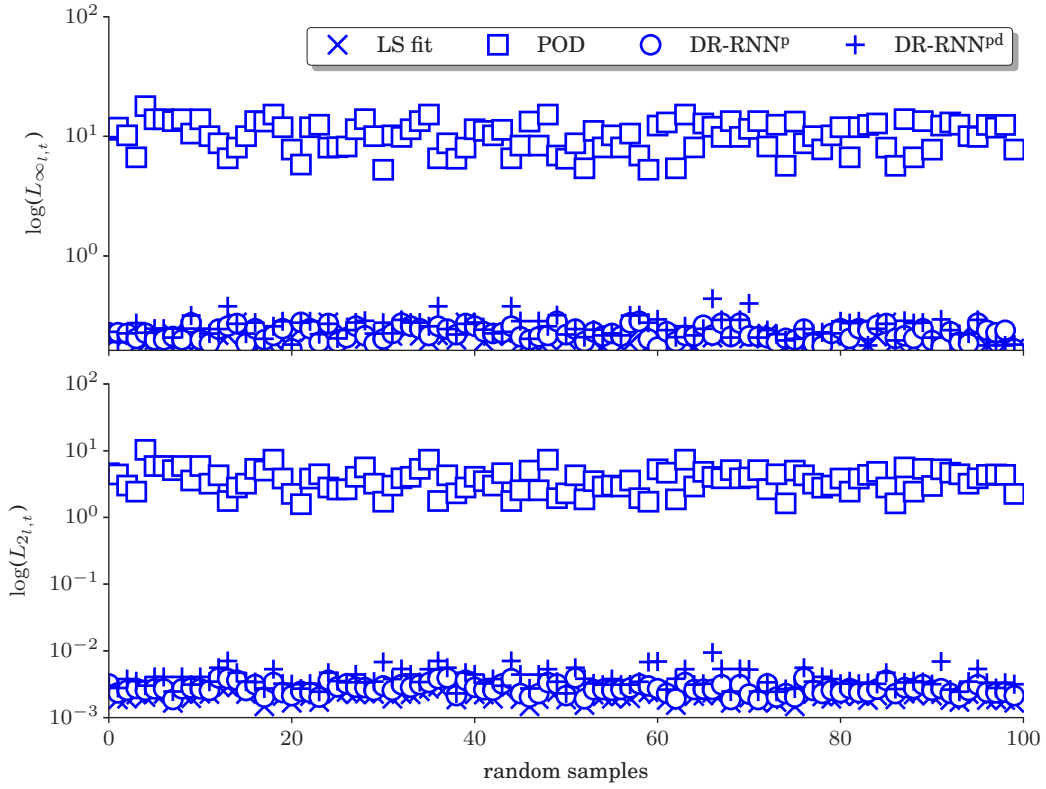


Figure 3.10: Comparison of $\log(L_{2_{l,t}})$ and $\log(L_{\infty_{l,t}})$ error estimators (Eq. (3.30)) at time = 0.3 PVI for test case 1. The number of POD basis used = 20.

Table 3.1: Performance chart of all the ROMs employed for test case 1. L_2^{rel} and $L_{2,\text{max}}^{\text{rel}}$ error estimators are defined in the Eq. (3.31). The number of POD basis used = 10 and 20.

Error	#Basis	Reduced Order Models			
		LS fit	POD	DR-RNN$^{\text{p}}$	DR-RNN$^{\text{pd}}$
L_2^{rel}	10	0.13	0.56	0.14	0.15
	20	0.10	2.7	0.11	0.13
$L_{2,\text{max}}^{\text{rel}}$	10	0.20	1.8	0.20	0.27
	20	0.17	5.8	0.19	0.26

We further list in Table 3.1, the L_2^{rel} and $L_{2,\text{max}}^{\text{rel}}$ errors for the saturation field. From Table 3.1, we can see that the approximation errors obtained from DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ have the same order of magnitude as the least–squares (best approximation) errors. Further, in Table 3.1, the approximation errors obtained from all ROMs except POD reduced model decreases when we increase the number of POD basis. These results conform with the decay of singular values of the saturation snapshot matrix. In Table 3.1, the approximation errors obtained from POD reduced model are at least an order of magnitude larger than other methods. Also, we observe that POD reduced model results might be worst when we include more basis functions. These results conform with the results presented in [67], where it was shown that selecting large number of basis vectors based on singular values may not lead to stable POD-Galerkin reduced model. Further, it was presented in [67] that the relation between the stability property of POD-Galerkin reduced model and the number of basis vectors used in POD-Galerkin projection is somewhat random and that the use of more POD basis vectors do not necessarily lead to improved stability.

3.5.6 Numerical test case 2

In this test case, the boundary conditions are set to no flow boundary conditions on the two sides aligned in the horizontal direction (top and bottom). Water is injected from the left side of the domain boundary and fluids are produced from the right side boundary of the domain. The total inflow rate from the left side is

set to 0.05 and the total outflow rate from the right side to 0.05 as the problem is incompressible. Similar to test case 1, we evaluate all the ROMs for two different number of saturation POD basis functions ($r = 10, 20$). Also, we fix the number of POD basis for the pressure state vector to 5. Figure 3.11 shows the setup for test case 2 and the corresponding singular values of the snapshot matrices $\mathbf{X}_p$, $\mathbf{X}_s$, and $\mathbf{X}_f$.

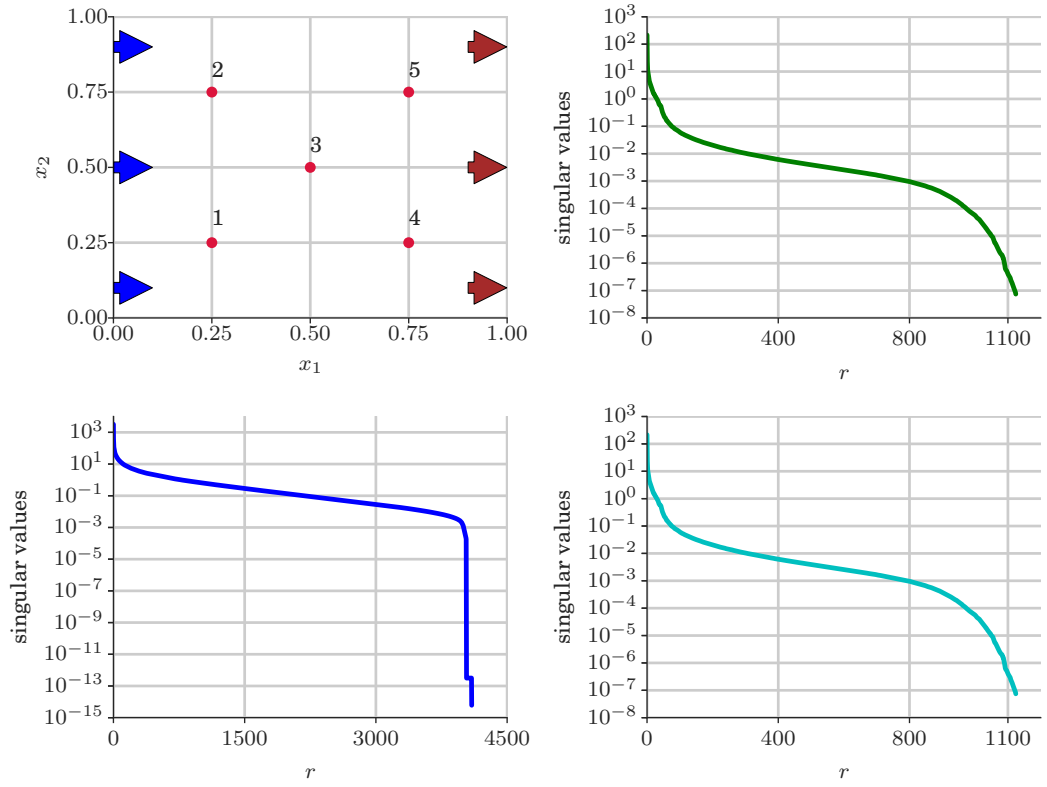


Figure 3.11: Top Left: Computational porous media domain in test case 2. The blue arrows in the left side corresponds to the injection of water and the brown arrows in the right side corresponds to the production of oil and water. The red dots represented in numbers from 1 to 5 corresponds to the locations where the PDF and the water saturation are investigated. Top Right: Singular values of the pressure snapshot matrix $\mathbf{X}_p$. Bottom Left: Singular values of the saturation snapshot matrix $\mathbf{X}_s$. Bottom Right: Singular values of the nonlinear function snapshot matrix $\mathbf{X}_f$

Figure 3.12 shows the time plot of mean water saturation obtained from all the ROMs and from the full model. The display settings in Figure 3.12 are the same as defined in Figure 3.4. In Figure 3.12, we can see that all the results obtained from DR-RNN$^{\text{p}}$, DR-RNN$^{\text{pd}}$, and the LS fit (the best approximation) closely approximates the full model whereas POD reduced model yields extremely inaccurate results regardless of the number of utilized POD basis.

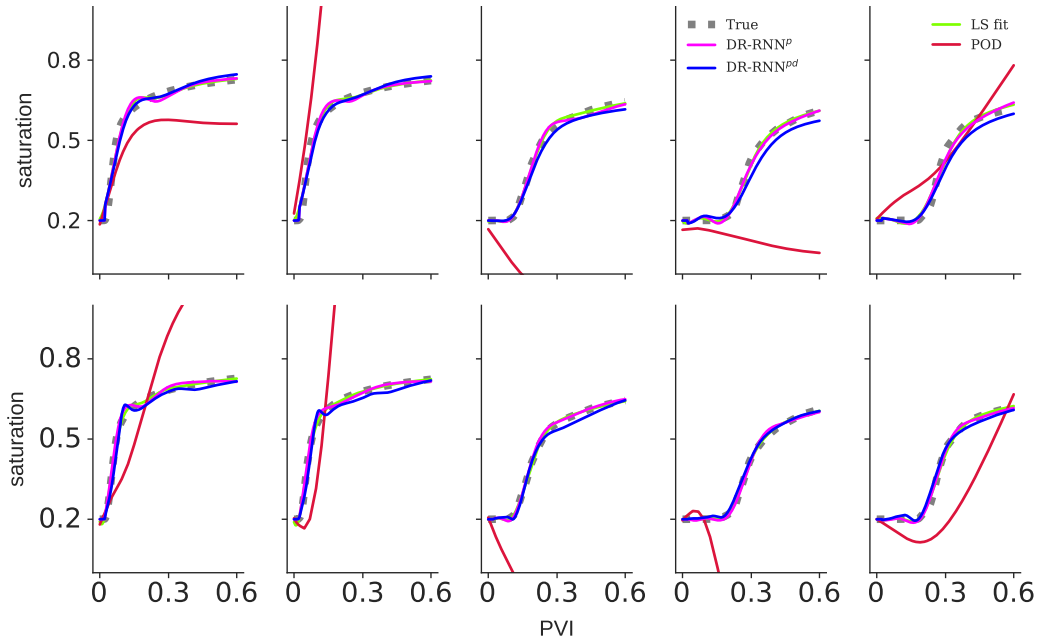


Figure 3.12: Time plots of mean water saturation obtained from all the ROMs and the full-order model in test case 2. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20. The plots in each row are arranged as per the numerical notation of the spatial points plotted in Figure 3.11.

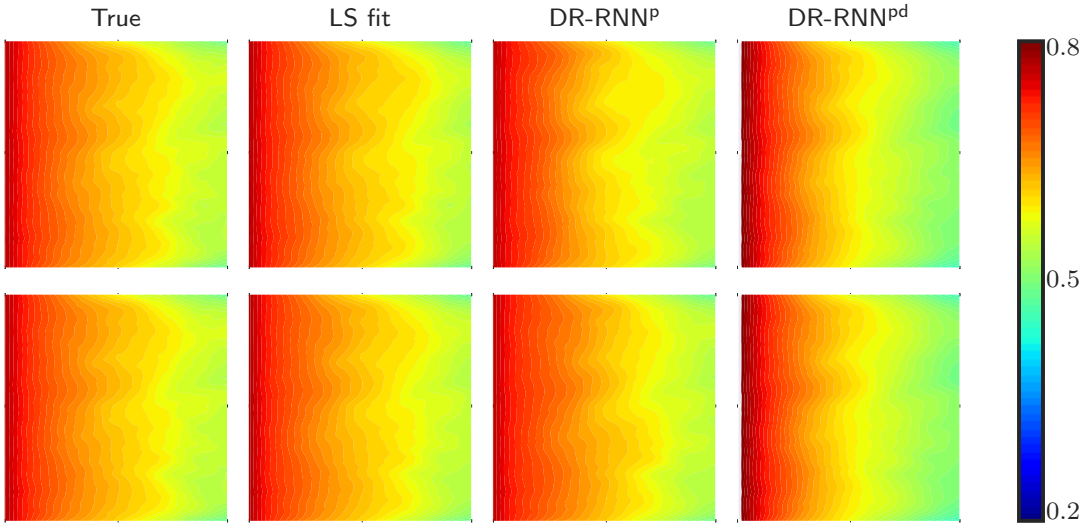


Figure 3.13: Comparison of mean water saturation field at time = 0.4 PVI for test case 2. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20.

Figures 3.13, 3.14, and 3.15 shows the results for the mean and standard deviation of the saturation field at 0.4 PVI. From these figures, we can conclude that all the plots obtained from DR-RNN ROMs are almost indistinguishable from the LS fit (the best approximation) results, whereas the plots obtained from POD reduced

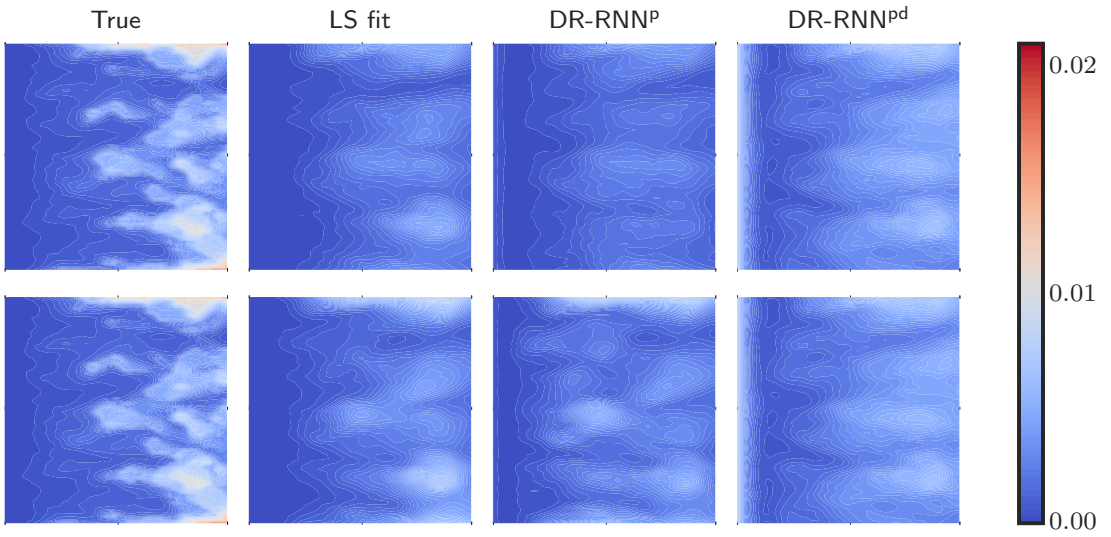


Figure 3.14: Comparison of standard deviation of the water saturation field at time = 0.4 PVI for test case 2. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20.

model (Figure 3.15) exhibit significant discrepancy when compared to the plots shown in Figure 3.13. Again, we note that the white spots displayed in Figure 3.15 are the regions whose values are out of the limits marked in the respective colorbar.

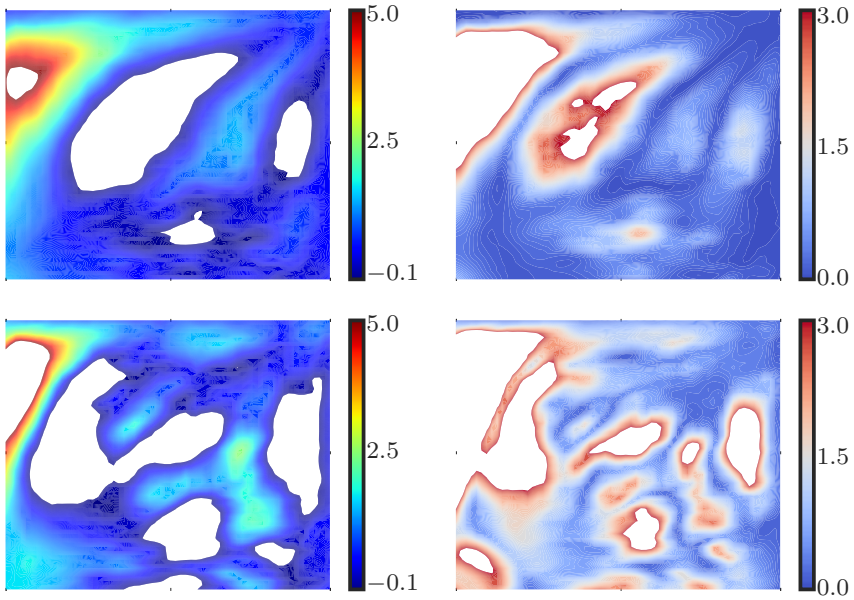


Figure 3.15: Plot of saturation mean and standard deviation of the water saturation field at time = 0.4 PVI obtained from the POD reduced model for test case 2. Left: saturation mean. Right: standard deviation. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20.

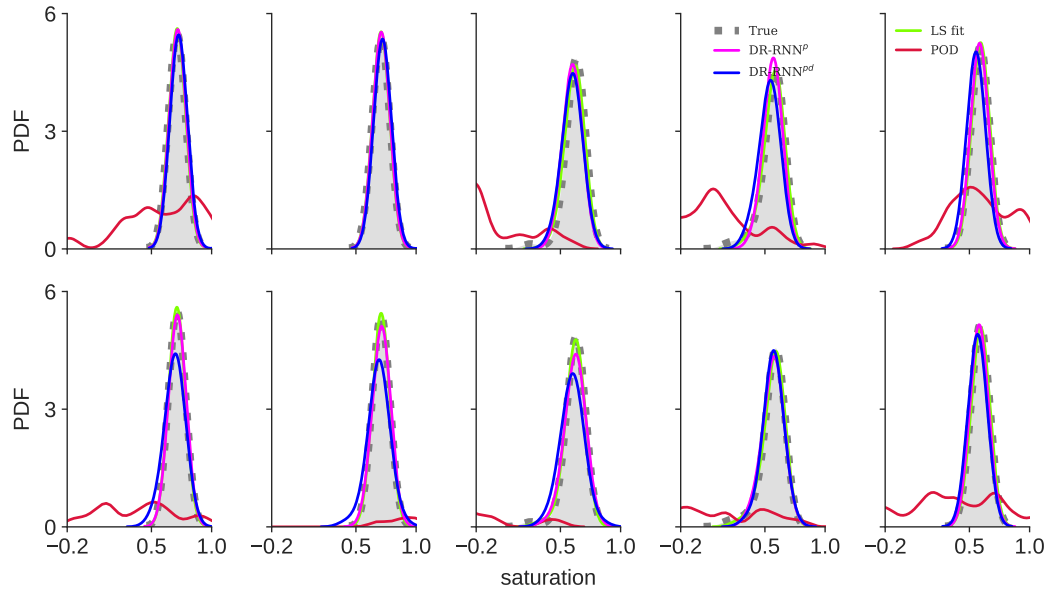


Figure 3.16: Comparison of kernel density estimated probability density function (PDF) at time = 0.4 PVI obtained from all ROMs w.r.t. true PDF obtained from the full-order model for test case 2. Top Row: number of POD basis used = 10. Bottom Row: number of POD basis used = 20. The plots in each row are arranged as per the numerical notation of the spatial points plotted in Figure 3.11.

Figure 3.16 compares the saturation PDF estimated from the ensemble of numerical solutions obtained from all the ROMs and the full model. The plotted results show that DR-RNN$^{\text{p}}$, DR-RNN$^{\text{pd}}$ predictions are nearly indistinguishable from the plots obtained from the full model and are very close to the best possible approximation using LS fit. Further, Figure 3.16 shows that all the saturation PDFs obtained from full model are uni-modal distribution. Similar to test case 1, POD reduced model yields inaccurate approximation of the saturation PDFs.

Table 3.2: Performance chart of all the ROMs employed for test case 2. L_2^{rel} and $L_{2,\text{max}}^{\text{rel}}$ error estimators are defined in the Eq. (3.31). The number of POD basis used = 10 and 20.

Error	#Basis	Reduced Order Models			
		LS fit	POD	DR-RNN$^{\text{p}}$	DR-RNN$^{\text{pd}}$
L_2^{rel}	10	0.09	1.30	0.10	0.12
	20	0.07	2.05	0.08	0.10
$L_{2,\text{max}}^{\text{rel}}$	10	0.19	3.5	0.21	0.22
	20	0.16	6.2	0.18	0.22

We further list in Table 3.2, the error metrics L_2^{rel} and $L_{2,\text{max}}^{\text{rel}}$ for the saturation

fields. From Table 3.2, we can see that the approximation errors obtained from DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ are almost close to the least–squares (best approximation) approximation errors. However, the POD reduced model yields very inaccurate results due to numerical instabilities.

3.6 Conclusion

In this work, we extended the DR-RNN introduced in [104] into nonlinear multiphase flow problem with distributed uncertain parameters. In this extended formulation, DR-RNN based on the reduced residual obtained from POD-DEIM reduced model is used to construct the reduced order model termed DR-RNN$^{\text{pd}}$. We evaluated the proposed DR-RNN$^{\text{pd}}$ on two forward uncertainty quantification problems involving two-phase flow in subsurface porous media. The uncertainty parameter is the permeability field modeled as log-normal distribution. In the two test cases, full order model and ROMs are solved for 2000 random permeability realizations to estimate an ensemble based statistics using Monte-Carlo method. full model and POD reduced model used implicit time stepping method as the time step size violates the numerical stability condition. However, DR-RNN$^{\text{pd}}$ architecture employs explicit time stepping procedure for the same step size used in full model and POD reduced model. Hence, DR-RNN$^{\text{pd}}$ had a limited computational complexity $\mathcal{O}(K \times r^2)$ instead of $\mathcal{O}(p \times r^3)$ per saturation update, where r is the dimension of the POD reduced model, $K \ll p$ is the number of stacked network layers in DR-RNN and p is the average number of Newton iterations used in the standard POD-DEIM reduced model. The obtained numerical results shows that DR-RNN$^{\text{pd}}$ provides accurate and stable approximations of the full model in comparison to the standard POD reduced model.

Future work should consider the development of accurate and stable DR-RNN$^{\text{pd}}$ for UQ tasks involving subsurface flow simulations with the additional effects including the capillary pressure, compressibility, and the gravitational effects. In addition, it will be of interest to explore the applicability of DR-RNN$^{\text{pd}}$ for UQ tasks with the permeability fields that has randomly oriented channels or barriers. The use of

DR-RNN^{pd} for history matching [48, 49], where we minimize the mismatch between simulated and field observation data by adjusting the geological model parameters is also expected to show significant reduction of the computational cost.

Chapter 4

Multi-Fidelity Framework With Multi-Level Monte Carlo Subsurface Flow Simulation [1]

4.1 Introduction

Effective propagation of uncertainties through nonlinear dynamical systems have become an essential task for model based engineering applications (e.g. water resources management, petroleum reservoir management) [115, 49, 84]. There are many possible sources of uncertainties in the input of multi-phase porous media flow model such as material properties (e.g. permeability, and porosity), boundary conditions, and geometrical information of the simulated domain. In this work, we focus on the canonical problem of uncertainty propagation in subsurface flow models due to the stochastic model inputs mainly the spatially distributed hydraulic conductivity field. In this setting, the high-fidelity model outputs (quantities of interest (QoI)) are usually defined as a time series of transport variables at a selected grid blocks (e.g. well locations) in the porous media domain. The propagation of uncertainties through multi-phase porous media flow models remains challenging because of high dimensionality of input parameter space (e.g. heterogeneous

[1] The contents of this chapter has been planned to submit to Frontiers in Environmental Science journal.

permeability) and the non-polynomial model nonlinearities [48, 49]. For this class of problems, probabilistic techniques, including stochastic Galerkin [60, 127], and stochastic collocation methods [10, 41] have limited applicability despite they are computationally very effective for quasi-linear flow models with the small number of random variables [91, 93].

One viable option to handle such situations is the Monte Carlo method (MC) where repeated evaluations of the high-fidelity flow models using different instantiations of the random input is performed. The output of these simulations is post-processed for estimates of the desired statistics such as the mean and the variance of the QoI. Generally, the estimators of MC method are unbiased. However, since the accuracy of MC method is measured in terms of the estimator variance [63], the convergence rate of MC estimators towards the desired statistics scales as $\sqrt{N}$, where N is the number of random samples. Given this slow convergence rate of MC methods, MC method is computationally expensive since a large number of high fidelity simulation have to be performed to obtain a reasonably accurate statistical estimates for the QoI. One notable advantage of MC methods in-comparison to other techniques [91, 93] is the ease of implementation using black-box simulators. Also, the rate of convergence is independent of the dimensionality of the random model inputs.

In this work, in order to make use of aforementioned advantages of the MC method and to alleviate the slow convergence rate, we employ a variant of control variate method [105, 64] called Multi-Level Monte Carlo method [63, 64] which makes use of the correlation between the high-fidelity model output and a multi-level hierarchy of low-fidelity model outputs. The key aspect of MLMC method is the repartition of the computational cost between different hierarchical levels of models based on the number of samples required to decrease the variance at each level. More precisely, the MLMC method relies on the fact that increasing the number of samples reduce the variance at low levels and at high levels, the level variances are expected to be typically small and thus MLMC method incurs few expensive high-fidelity simulations [63, 103].

Similar to MLMC method, Multi-Fidelity Monte Carlo method method [105, 113] is an another control variate method which combines the outputs from an arbitrary number of low-fidelity models with the high-fidelity model in order to speedup the statistical estimation of the QoI. The key aspect of MFMC approach is the initial selection of low-fidelity models and the corresponding number of model runs for each model [105, 113]. Ng [105] proposed a multifidelity approach to reduce the cost of expensive objective functions in stochastic optimization problems by making use of inexpensive, low-fidelity models. Peherstorfer et al. [113] extended the MFMC method introduced in [105] to accelerate UQ tasks by making use of many number of low-fidelity models. Furthermore, MFMC method introduced in [105] can utilize low-fidelity models of any time, for example up-scaled models [43], POD reduced order models [17, 89, 5] and response surface based models [53] could be combined in the MFMC framework. Geraci et al. [58] solved an UQ task involving turbulent flow over a periodically arranged hills using MFMC method where Direct Numerical Simulation was used as a high-fidelity model and Reynolds Average Navier Stokes equation was used as a low-fidelity model. We refer the readers to read the paper by [114] for a complete review of MFMC method.

We now present a brief literature review of MLMC method as applied to uncertainty quantification (UQ) tasks. It appears Heinrich [72] was the one to first apply MLMC in the context of parametric integration. Kebaier et al. [85] then used similar ideas for a two-level Monte Carlo method to approximate weak solutions to stochastic differential equations in mathematical finance. Giles [62] extended the MLMC method to solve stochastic ordinary differential equations of Ito type. Barth et al. [13] and Cliffe et al. [36] introduced MLMC method for elliptic partial differential equations (PDEs) with stochastic coefficients. Abdulle et al. [3] applied MLMC method to solve elliptic PDEs in divergence form, where the coefficients are random with multiple scales. Mishra et al. [101] generalized MLMC method to nonlinear, scalar hyperbolic conservation laws with random initial data. Mishra et al. [102] extended the work of [101] for systems of nonlinear, hyperbolic conservation laws in several space dimensions. Geraci et al. [59] proposed a Multi-Level Multi-Fidelity

method in which the MLMC estimator is modified at each level to benefit from a level specific low-fidelity model.

In the context of fluid flow in porous media, Müller et al. [103] applied MLMC method for two phase transport simulations of an oil reservoir with uncertain heterogeneous permeability. Efendiev et al. [44] used mixed multi-scale finite element methods within the MLMC framework to speed up the computations involving multiphase flow and transport simulations. Efendiev et al. [45] coupled the generalized multi-scale finite element method with the Multi-Level Markov chain Monte Carlo method (MLMCMC), which sequentially screens the proposal with different levels of approximations and combines the samples at different levels to arrive at an accurate estimate. Elsakout et al. [47] demonstrated the performance of (MLMCMC) for uncertainty quantification tasks involving reservoir simulation with less computional cost in comparison to standard Markov Chain Monte Carlo method. Fagerlund et al. [51] combined selective refinement technique with the MLMC for estimating the sweep efficiency in a two-phase flow scenario where an absolute accuracy of failure probability in a magnitude 5 to 10 percent is required. Lu et al. [95] applied MLMC method for estimating cumulative distribution functions of QoI obtained from the numerical approximation of large-scale stochastic subsurface simulations. For a complete review of MLMC method, we refer the readers to the following papers by Giles [63, 64].

Historically, MLMC method constructs a hierarchy of coarse spatial and/or time discretization models as low-fidelity models. However, it is also possible to formulate a sequence of low-fidelity models utilizing projection based reduced order models [143, 136] of different dimensions. For example, Codina et al. [37] employed different reduced basis ROMs in the MLMC framework to estimate the statistical outputs of stochastic elliptic PDEs. In that work, the authors proposed an algorithm for optimally choosing both the dimensions of the reduced basis ROMs and the number of Monte Carlo samples at each level to achieve a given error tolerance.

In this manuscript, we propose a Multi-Fidelity-Multi-Level Monte Carlo (MFMLMC) method to address some of the limitations of standard MLMC method when

Galerkin projection based ROMs [37, 89, 5] are used as low-fidelity models. We first note that the variance and hence the mean square error of the standard MLMC estimator depends on the correlation between every two consecutive level ROMs. This requires the utilization of a large number of level with small differences in the number of dimensions between each two consecutive ROMs. Therefore, the standard MLMC estimator not only requires many levels of ROMs but also requires ROMs of high dimensions until high correlations with the high-fidelity models are achieved. Hence, the MLMC method involving ROMs obtained directly from high-fidelity model solution data like the one mentioned in [37] can significantly limit the performance of MLMC method. Second, Galerkin-projection ROMs like POD ROMs obtained from nonlinear high-fidelity model are subject to severe convergence and stability issues especially when the dimensions of the ROMs are much smaller than the dimensions of the high-fidelity model [137, 67, 24]. This severely limits the use of POD ROMs with low dimensions in MLMC framework and therefore, we cannot expect the reduction in computational complexity by orders of magnitude as a result of state variable's dimension reduction [84]. Third, MLMC method based on ROMs requires reconstruction of the high-fidelity model state variable for every sample at each level. Such reconstruction of high-fidelity model state variable involves a high dimensional matrix vector multiplication, and therefore employing ROMs in MLMC method can easily cause computational overheads. Fourth, the nonlinear integer optimization problem formulated in [38] does not guarantee to determine the optimal dimensions of the ROMs despite of additional computational complexity. Fifth, MLMC method benefits from the fact that the variance at high levels are typically small and thus requires only small number of expensive high-fidelity model outputs to achieve the desired mean square error. However, despite requiring only small number of expensive high-fidelity simulations, MLMC method optimization algorithm has to allocate a significant portion of the given computational budget to high level models. Therefore, within the remaining limited budget, MLMC algorithm is constrained to draw only a limited number of samples from low levels despite low level models being computationally cheap.

The pivotal ideas of the proposed MFML-MC method are detailed as follows. The first pivotal idea of the MFML-MC approach is to obtain a sequence of POD based approximations of the QoI directly from the training samples of the QoI. More precisely, we compute the optimal POD bases from the singular value decomposition of the snapshot matrix built from the training samples of the QoI. We then employ the computed POD bases in the least-squares reconstruction method to obtain a sequence of POD based approximations of the QoI (see section 4.4 for more details). We use these sequence of POD based approximations as low-fidelity models in MLMC algorithm of MFML-MC method. Since, QoI is much smaller than the state variables dimension, the basis vector dimension for the QoI POD is much smaller than when building a standard POD model. Therefore, building QoI POD instead of a full state variable POD enables the efficient extraction of high-level PODs at limited computational cost. The second pivotal idea is to build a ROM for the difference between every two consecutive level in the MLMC formulation. We utilize principal component analysis (PCA) to first perform dimensional reduction and then utilize machine learning approaches to learn the dynamic evolution of the extracted ROM. The third pivotal idea in the MFML-MC approach is to use MFMC method at each level so that the high-fidelity model is utilized to provide an unbiased estimator, while the low computational cost of the low-fidelity model is exploited to run a very large number of realizations to obtain a low variance estimator. The fourth pivotal idea is to utilize the extracted ROM mentioned in the step two as a low-fidelity model in the MFMC setup on every levels of MFML-MC method. We use use a variant of DR-RNN called Model-Free DR-RNN (MF-DR-RNN) to formulate level specific low-fidelity model in the MFMC setup on every levels. DR-RNN is a recurrent neural network whose architecture is inspired by the iterative line search methods where the parameters of the DR-RNN are optimized such that the residual of the nonlinear dynamical system is minimized [18, 130, 104]. In MF-DR-RNN, we first formulate the dynamics from the time series data of the unknown system using standard regression technique (e.g feed-forward Artificial Neural Network (ANN), Gradient Boosted Tree Regressor (GBTR) [55]). Then, we utilize the dynamics ob-

tained from the regression technique to formulate the architecture of MF-DR-RNN. To the best of our knowledge, this paper presents the first attempt to combine the features of MFMC method and MLMC method using machine learning techniques for UQ analysis of nonlinear dynamical systems representing multi-phase porous media flow with uncertainty in the permeability field.

The remaining of this chapter is organized as follows. In Section 2, multi-phase porous media flow problem is formulated. In Section 3, MC, MFMC, and MLMC methods are briefly explained. In section 4, MFML-MC method is introduced. In section 5, Numerical results for two subsurface multiphase porous media flow problems showing the performance of MFML-MC method are reported. Finally, in section 6, conclusions and perspectives are drawn.

4.2 Problem Formulation

We consider an immiscible two-phase (oil and water) flow in an incompressible porous media domain. The flow behaviour of oil and water in a porous media domain can be described by conservation of mass and Darcy's law for each phase [34, 1, 14]. Neglecting the effects of gravitation, capillary, and compressibility, and assuming the density ratio to be equal to one, Darcy's law for each phase can be described as

$$\mathbf{v}_\alpha = -\mathbf{K}\frac{k_{r\alpha}}{\mu_\alpha}\,\nabla p \tag{4.1}$$

where the subscript $\alpha = w$ denotes the water phase, the subscript $\alpha = o$ denotes the oil phase, $\mathbf{v}_\alpha$ is the phase velocity, p is the global pressure, $\mathbf{K}$ is the absolute permeability tensor, $k_{r\alpha}$ is the relative permeability of phase α, μ_α is the viscosity of phase α [34, 1, 14]. The phase relative permeabilities $k_{r\alpha}$ models the interactions between the two phases and usually $k_{r\alpha}$ is described as a function of phase saturation (volume of phase α in a given pore space of the porous media domain) [1].

The total conservation of mass can be expressed in terms of incompressibility condition that takes the form

$$\nabla \cdot \mathbf{v} = q \tag{4.2}$$

$\mathbf{v} = \mathbf{v}_o + \mathbf{v}_w$ is the total velocity vector and q is the total source and sink term. We can combine the equation of Darcy's law for each phase (Eq. (4.1)) and the conservation of mass (Eq. (4.2) to derive equations for global pressure and water saturation:

$$\begin{aligned} \nabla \cdot \mathbf{K}\lambda \ \nabla p &= q \\ \phi \frac{\partial s_w}{\partial t} - \nabla \cdot (f_w \mathbf{v}) + q_w &= 0 \end{aligned} \tag{4.3}$$

where $\lambda = \lambda_w + \lambda_o$ is the total mobility, $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$ is the phase mobility, $f_w = \lambda_w/(\lambda_w + \lambda_o)$ is termed as the fractional flow function for the water phase and with the constraint $s_w + s_o = 1$. In the rest of the manuscript, we use s in place of s_w to denote water saturation.

In this problem, we consider Eq.(4.3) as the high-fidelity model and we solve Eq. (4.3) for pressure and saturation using sequential formulation [1] where we solve for pressure first and then solve for the water saturation. We use finite volume method to disctretize the spatial derivatives of Eq.(4.3) in a spatial domain of n grid blocks. The resultant nonlinear dynamical system for updating saturation field takes the form

$$\frac{d\mathbf{y}_s}{dt} + \mathbf{B}(\mathbf{v}) \ \mathbf{f}_w(\mathbf{y}_s) = \mathbf{d} \tag{4.4}$$

where $\mathbf{y}_s \in \mathbb{R}^n$ is the high-fidelity model state variable and each component of $\mathbf{y}_s$ is the water saturation value at the ith grid block, $\mathbf{B} \in \mathbb{R}^{n \times n}$, $\mathbf{d} \in \mathbb{R}^n$, $\mathbf{v}$ is the total velocity vector. We then use Euler implicit time stepping method to transform the continuous dynamical system (Eq. (4.4)) to discrete dynamical system that takes the form

$$\mathbf{y}_s(t+1) = \mathbf{y}_s(t) - \mathbf{B}(\mathbf{v}(t)) \ \mathbf{f}_w(\mathbf{y}_s(t+1)) \ \Delta_t - \mathbf{d} \ \Delta_t \tag{4.5}$$

The QoI is defined as $\mathbf{u}(t) \in \mathbb{R}^m$, where $u_i = y_s(x_i, y_i)$, $i = 1 \cdots m \ll n$ at specific time steps (say $t = 10, 20, \cdots 200$). In the following, we use $\mathbf{u}$ in place of $\mathbf{u}(t)$ to simplify the notation and we are interested in the first moment estimate (i.e. mean) of $\mathbf{u}$. The grid points of interest (x_i, y_i) $i = 1 \cdots m$ can be a set of arbitrary user specific spatial locations. For example, a set of grid points where injectors and producers are located.

4.3 Standard Multi-Fidelity Monte Carlo and Multi-level Monte-Carlo method (current state of the art)

Let $\mathbf{x}$ be a realization of the input random vector $\mathbf{X}(\omega)$, $\omega \in \Omega$ where Ω is the sample space and the quantity of interest be the expectation of the random variable $\mathbf{u}$. The standard Monte Carlo method estimates the expectation $\mathbb{E}[\mathbf{u}]$ of the random variable $\mathbf{u}$ as

$$\hat{\mathbf{u}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{u}^i \tag{4.6}$$

where $\hat{\mathbf{u}}$ is the estimator of $\mathbb{E}[\mathbf{u}]$, $\mathbf{u}^i = \mathbf{u}(\mathbf{x}_i)$, and N is the number of realizations of the model output. As per the law of large numbers (Central Limit Theorem) [64], a sample base estimate of the expectation $\mathbb{E}[\mathbf{u}]$ introduces sampling error (mean square error) defined as

$$\epsilon = \mathbb{V}\text{ar}(\hat{\mathbf{u}}) = \frac{1}{N}\mathbb{V}\text{ar}(\mathbf{u}) \tag{4.7}$$

where $\mathbb{V}\text{ar}(\mathbf{u})$ is the variance of $\mathbf{u}$. As $\sqrt{\epsilon}$ known as standard error scales with $\frac{1}{\sqrt{N}}$ for a constant $\mathbb{V}\text{ar}(\mathbf{u})$, MC simulations are computationally prohibitive because of the slow convergence rate. One way to achieve a lower ϵ is to reduce the numerator in Eq. (4.7) [105].

Control variate is a variance reduction technique which uses alternative estimator for $\mathbb{E}[\mathbf{u}]$, $\mathbf{u} \in \mathbb{R}$ that takes the form

$$\hat{\mathbf{u}}^{cv} = \hat{\mathbf{u}} + \beta\ (\hat{\mathbf{v}} - \mathbb{E}[\mathbf{v}]) \tag{4.8}$$

where $\mathbf{v}(\mathbf{x}) \in \mathbb{R}$ is an auxiliary random variable. The estimator $\hat{\mathbf{u}}^{cv}$ is an unbiased estimator of $\mathbb{E}[\mathbf{u}]$ with variance defined as [105]

$$\mathbb{V}\text{ar}(\hat{\mathbf{u}}^{cv}) = \mathbb{V}\text{ar}(\hat{\mathbf{u}})\ (1 - \rho^2) \tag{4.9}$$

where ρ is the correlation between $\mathbf{u}(\mathbf{x})$ and $\mathbf{v}(\mathbf{x})$. Since ρ^2 lies between 0 and 1, $\mathbb{V}\text{ar}(\hat{\mathbf{u}}^{cv})$ is always less than $\mathbb{V}\text{ar}(\hat{\mathbf{u}})$. For UQ tasks where the QoI is governed by

partial differential equations, $\mathbf{u}(\mathbf{x})$ is obtained from a high-fidelity model output and $\mathbf{v}(\mathbf{x})$ is generally obtained from a low-fidelity model output. In general, we do not know exactly $\mathbb{E}[\mathbf{v}]$ and we have to use a more accurate estimate of $\mathbb{E}[\mathbf{v}]$. For example, Ng [105] replaced $\mathbb{E}[\mathbf{v}]$ in Eq. (4.8) by $\hat{\mathbf{v}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{v}(\mathbf{x}_i)$, where $M \gg M_{\text{HF}}$, and M_{HF} is the number of high-fidelity model samples. Furthermore, it was proved in [105] that for a fixed computational budget p, a perfectly correlated low-fidelity model is not the only condition for variance reduction over the standard MC estimator but the low-fidelity model must also be cheaper to evaluate than the high-fidelity model.

The potential limitation in the aforementioned multi-fidelity estimator [105] is that it repartition the given computational budget p between the high-fidelity model and only a single low-fidelity model such that the mean square error of the estimator is minimized. In order to allow an arbitrary number of low-fidelity models into the control variate method, Peherstorfer et al. [113] extended the multi-fidelity approach introduced in [105]. Multi-Fidelity Monte Carlo method introduced in [113] formulated an optimization problem that uses an arbitrary number of low-fidelity models to derive an unbiased MFMC estimator of $\mathbb{E}[\mathbf{u}]$ that takes the form

$$\hat{\mathbf{u}}^{mf} = \hat{\mathbf{u}} + \beta^1\ (\hat{\mathbf{v}}^1 - \hat{\mathbf{u}}) + \sum_{i=2}^{I}\beta^i\ (\hat{\mathbf{v}}^i - \hat{\mathbf{v}}^{i-1}) \tag{4.10}$$

where $\mathbf{v}^1 \cdots \mathbf{v}^I \in \mathbb{R}$ are auxiliary random variables obtained from I number of different low-fidelity models, $\hat{\mathbf{v}}^i$ estimates the expectation $\mathbb{E}[\mathbf{v}^i]$ using M_i samples of low-fidelity model i, $\beta^1 \cdots \beta^I \in \mathbb{R}$ are the coefficients. The low-fidelity model i uses $\mathbf{x}_1 \cdots \mathbf{x}_{M_i}$ realizations of the input random vector $\mathbf{X}(\omega)$ to estimate $\hat{\mathbf{v}}^i$, whereas the low-fidelity model $i-1$ uses only the first M_{i-1} realizations of $\mathbf{X}(\omega)$ to estimate $\hat{\mathbf{v}}^{i-1}$. Therefore the two consecutive estimators $\hat{\mathbf{v}}^i$ and $\hat{\mathbf{v}}^{i-1}$ are dependent for all $i = 1 \cdots I$. The cost of the MFMC estimator is $C(\hat{\mathbf{u}}^{mf}) = \sum_{i=1}^{I} C_i \cdot M_i + C_{\text{HF}} \cdot M_{\text{HF}}$, where C_{HF} is the cost of evaluating a high-fidelity model, and C_i is the cost of evaluating a low-fidelity model i for all i for $i = 1 \cdots I$. In [113], an optimization problem was formulated to select optimal values for the number of samples $\{M^*_{\text{HF}}, M^*_1 \cdots M^*_I\}$, and for the coefficients $\{\beta^{1^*} \cdots \beta^{I^*}\}$ such that the mean square error of the MFMC estimator is lower than the Monte Carlo estimator for a fixed computational budget.

The multi-level idea is an another extension of the control variate approach in which a sequence of low-fidelity models at different levels ($\mathbf{v}_i \in \mathbb{R}^m$ with $i = 1 \cdots I$) is used to evaluate an approximate statistics of $\mathbf{u}$. First, let the index i encodes the accuracy of $\mathbf{v}_i$ with respect to the true solution $\mathbf{u} \in \mathbb{R}^m$. This means, as i is increased, the accuracy of $\mathbf{v}_i$ is refined to approximate $\mathbf{u}$. Consequently, $\mathbf{u}$ can be written as a telescopic sum in terms of $\mathbf{v}_i$ with $i = 1 \cdots I$, that takes the form [103]

$$\mathbf{u} = \mathbf{v}_1 - \mathbf{v}_0 + \mathbf{v}_2 - \mathbf{v}_1 + \cdots + \mathbf{v}_I - \mathbf{v}_{I-1} + \mathbf{u} - \mathbf{v}_I = \sum_{i=0}^{I} \mathbf{Y}_i \tag{4.11}$$

where $\mathbf{Y}_i = \mathbf{v}_{i+1} - \mathbf{v}_i$ with $i = 0 \cdots I-1$, $\mathbf{Y}_I = \mathbf{u} - \mathbf{v}_I$, and we set $\mathbf{v}_0 = 0$. Exploiting the linearity of the expected value operator $\mathbb{E}$, the expected value $\mathbb{E}[\mathbf{u}]$ defined in Eq. (4.11) can be written as

$$\mathbb{E}[\mathbf{u}] = \sum_{i=0}^{I} \mathbb{E}[\mathbf{Y}_i] \tag{4.12}$$

The MLMC estimator for the expected value of $\mathbf{u}$ is obtained by replacing the expected values on the right hand side of Eq. (4.12) by ensemble averages and is defined as

$$\hat{\mathbf{u}}^{ml} = \sum_{i=0}^{I} \hat{\mathbf{Y}}_i = \sum_{i=0}^{I} \frac{1}{M_i} \sum_{j=1}^{M_i} \mathbf{Y}_i^j \tag{4.13}$$

The mean square error (mse) of MLMC estimator $\hat{\mathbf{u}}^{ml}$ is derived as

$$\epsilon^{ml} = \sum_{i=0}^{I} \mathbb{V}\mathrm{ar}(\hat{\mathbf{Y}}_i) = \sum_{i=0}^{I} \frac{1}{M_i} \mathbb{V}\mathrm{ar}(\mathbf{Y}_i) \tag{4.14}$$

It is evident from Eq. (4.14) that the mse (ϵ^{ml}) of MLMC estimator is sum of several smaller contributions $\frac{1}{M_i}\mathbb{V}\mathrm{ar}(\mathbf{Y}_i)$ with $i = 0 \cdots I$.

The MLMC method is mainly based on the fact that $\frac{1}{M_i}\mathbb{V}\mathrm{ar}(\mathbf{Y}_i)$ at low levels are reduced by increasing number of samples (M_i) as low level samples are computed at low computational cost. At high levels, the level variances $\mathbb{V}\mathrm{ar}(\mathbf{Y}_i)$ are expected to be typically small, thus M_i can be small and hence MLMC method incurs few expensive high-fidelity model simulations. In summary, MLMC method relies on

the following variance hierarchy:

$$\mathbb{V}\text{ar}(\mathbf{Y}_0) > \mathbb{V}\text{ar}(\mathbf{Y}_1) > \mathbb{V}\text{ar}(\mathbf{Y}_2) > \cdots > \mathbb{V}\text{ar}(\mathbf{Y}_I) \tag{4.15}$$

and also expects $C_0 < C_1 < C_2 < \cdots < C_I$, where C_i is the computational cost to compute one sample of $\mathbf{Y}_i$. In MLMC method, the optimal values for the number of samples $\{M_0^* \cdots M_I^*\}$ are computed by solving a constrained minimization problem where the cost function to be minimized is the total computational cost ($\sum_{i=0}^{I} C_i \cdot M_i$) of the MLMC method and constraint is set by fixing ϵ^{ml} to a specific value (say $\frac{\epsilon^2}{2}$) [103, 59]. The optimal values for the number of samples are expressed as

$$M_i^* = \frac{2}{\epsilon^2} \left[\sum_{j=0}^{I} \sqrt{C_j \cdot \mathbb{V}\text{ar}(\mathbf{Y}_j)} \right] \sqrt{\frac{\mathbb{V}\text{ar}(\mathbf{Y}_i)}{C_i}} \qquad i = 0, \cdots, I \tag{4.16}$$

Although MLMC in general refer to control variate method with a sequence of I geometrical levels (mesh discretization levels), it can also be utilized with a sequence of I reduced basis models [38] or POD basis models. More specifically, a sequence of POD basis models can be employed as sequence of low-fidelity models $\mathbf{v}_1 \cdots \mathbf{v}_I$.

A practical implementation of the MLMC algorithm is the following [103]

1. Fix a sequence of levels based on grid resolutions or POD basis $i = 1 \cdots I$.
2. Fix a number of offline samples M_o and fix a threshold for the estimated standard error.
3. Perform M_o samples of high fidelity simulations.
4. If POD basis models, Derive I number of POD basis models.
5. Compute M_o samples of $\mathbf{Y}_i$ on every level.
6. Solve the optimization [103] problem to estimate M_i samples of $\mathbf{Y}_i$ with $i = 0 \cdots I$.
7. Update the estimates for $\mathbb{E}[\mathbf{Y}_i]$, $\mathbb{V}\text{ar}(\mathbf{Y}_i)$, and C_i on every level.
8. Compute and update the required number of samples M_i on each level.

9. On every level, if the updated M_i is more than the number of samples already computed, then add an additional sample of $\mathbf{Y}_i$ and continue with step 6. If no level requires an additional sample, then quit.

4.4 New contribution to Multi-Fidelity and Multi-level Monte Carlo method

In this section, we present a novel variance reduction method called Multi-Fidelity-Multi-Level Monte Carlo (MFML-MC) method addressing the limiting facts observed in the standard MLMC method with Galerkin projection based ROMs (see section 4.1 for more details). In MFML-MC method, we formulate a MLMC framework with I levels and then apply the techniques of MFMC method on every level of MLMC framework. Figure 4.6 displays the outline of the MFML-MC method and its detailed formulation is described as five steps in the rest of this section.

The first step of MFML-MC method is to formulate a sequence of POD approximations of $\mathbf{u}(t)$ in order to be utilized as low-fidelity models in MLMC framework. We derive this sequence of POD approximations of $\mathbf{u}(t)$ directly from the training data. More precisely, we first compute the optimal POD bases from the singular value decomposition (SVD) of the snapshot matrix $\mathbf{X}_u = \big((\mathbf{u}_1 \ \dots \ \mathbf{u}_T)^1 \ \dots \ (\mathbf{u}_1 \ \dots \ \mathbf{u}_T)^L\big)$, where T denotes the number of time steps and L denotes the number of samples corresponding to different realizations of the stochastic input parameters. The SVD of $\mathbf{X}_u$ is expressed as [84]

$$\mathbf{X}_u = \mathbf{U}_u \ \Sigma_u \ \mathbf{W}_u \tag{4.17}$$

where $\mathbf{U}_u \in \mathbb{R}^{m \times m}$ is the left singular matrix, $(\sigma_1 > \sigma_2 > \sigma_3 > \cdots \ \sigma_m \geq 0)$ are the singular values of the snapshot matrix $\mathbf{X}_u$. The QoI $\mathbf{u}$ is now optimally expressed as

$$\mathbf{u} \approx \mathbf{v} = \mathbf{U}_u^r \ \tilde{\mathbf{u}} = \mathbf{U}_u^r \ (\mathbf{U}_u^{r\top} \ \mathbf{u}) \tag{4.18}$$

where $\tilde{\mathbf{u}} \in \mathbb{R}^r$ is the reduced representation of $\mathbf{u}$, $\mathbf{U}_u^r \in \mathcal{R}^{m \times r}$ is the orthonormal matrix containing r orthonormal basis vectors in its columns, and $\mathbf{v}$ is the POD

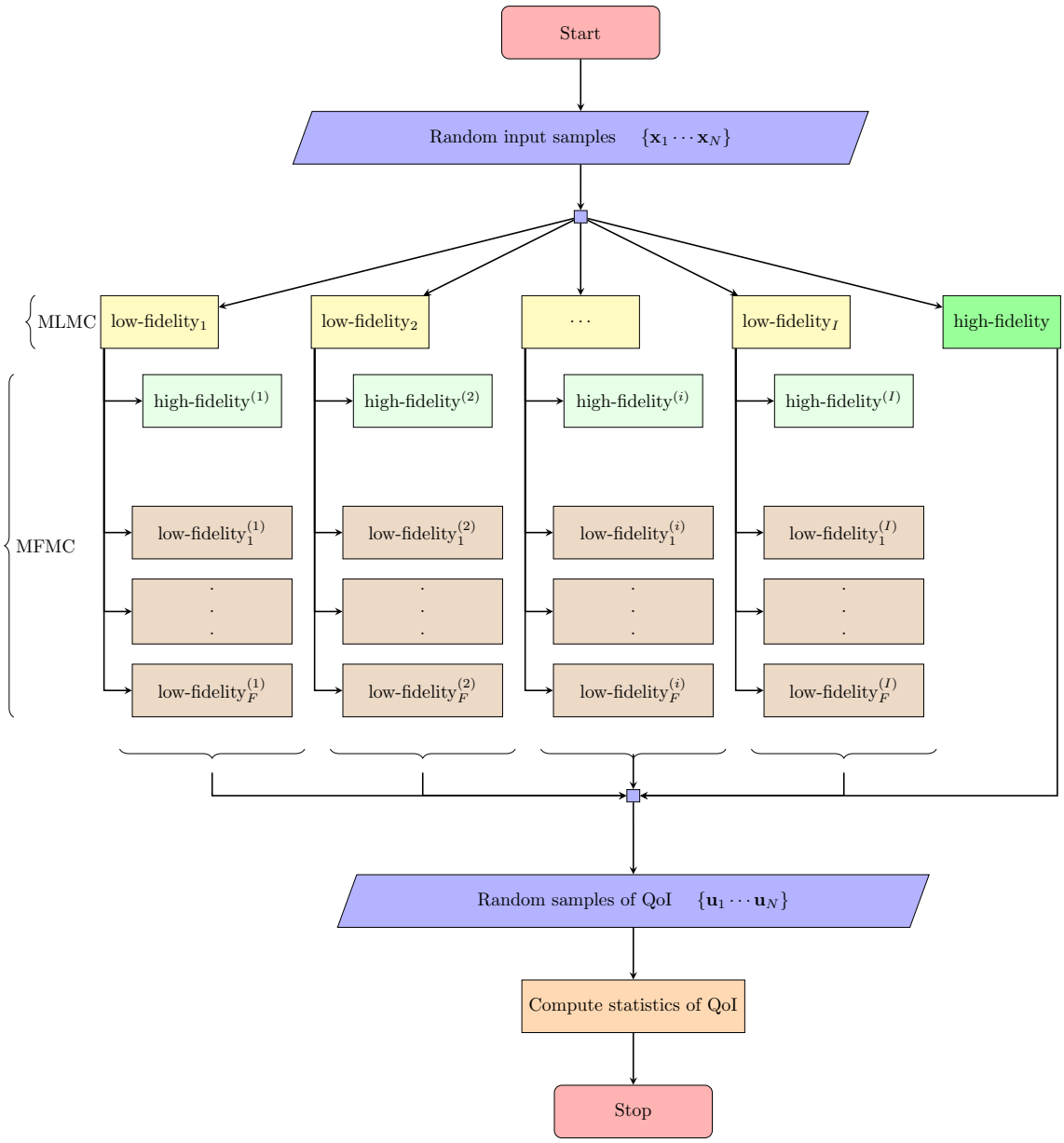


Figure 4.1: Outline of the (MFML-MC) method described in section 4.4. The low-fidelity$_i$ (yellow color) denotes low-fidelity model i $(i = 1, 2, \cdots I)$ in MLMC setup. The low-fidelity$_f^{(i)}$ (brown color) denotes low-fidelity f in MFMC method formulated in the ith MLMC setup. QoI denotes the quantity of interest or outputs of interest.

approximations of $\mathbf{u}$ obtained from least-squares reconstruction method (Eq. (4.18)). Please note that the dimension m of the basis vector in $\mathbf{U}_u$ is much smaller than n (the number of grid points). In this approach, $\mathbf{v}_i$ is ith level POD approximation of $\mathbf{u}$ obtained by using r_i pod basis vectors in Eq. (4.18), for all $i = 1 \cdots I$. Similarly, $\mathbf{X}_{v_i} = \left((\mathbf{v}_{i_1} \ \ldots \ \mathbf{v}_{i_T})^1 \ \ldots \ (\mathbf{v}_{i_1} \ \ldots \ \mathbf{v}_{i_T})^L\right)$ is the the ith level POD approximation of $\mathbf{X}_u$, for all $i = 1 \cdots I$. The associated error termed as least–squares errors in

approximating $\mathbf{u}$ by $\mathbf{v}_i$ using only r_i basis vectors is given by [96, 17]

$$\varepsilon = \|\mathbf{u} - \mathbf{v}_i\|_2 = \sum_{i=r_i+1}^{m} \sigma_i \tag{4.19}$$

where $\|\cdot\|_2$ denotes Euclidean norm as defined in Eq. (2.18) chapter 2.

The second step of MFML-MC method is to compute the training samples of $\mathbf{Y}_i = \mathbf{v}_{i+1} - \mathbf{v}_i$ over all levels in MLMC framework (see Eq. (4.11)). The training samples of $\mathbf{Y}_i$, $\mathbf{X}_{Y_i} = \big((\mathbf{Y}_{i_1} \ \dots \ \mathbf{Y}_{i_T})^1 \ \dots \ (\mathbf{Y}_{i_1} \ \dots \ \mathbf{Y}_{i_T})^L\big)$ are obtained from $\mathbf{X}_{v_{i+1}}$ and $\mathbf{X}_{v_i}$ by computing their difference (i.e., $\mathbf{X}_{Y_i} = \mathbf{X}_{v_{i+1}} - \mathbf{X}_{v_i}$) for all $i = 0 \cdots I-1$.

The third step of MFML-MC method is to compute the reduced representation of $\mathbf{Y}_i$ over all levels in MLMC framework. More precisely, we compute the reduced representation of $\mathbf{Y}_i$ from the optimal POD bases of $\mathbf{Y}_i$ which are obtained from the SVD of the snapshot matrix $\mathbf{X}_{Y_i}$. The SVD of $\mathbf{X}_{Y_i}$ is expressed as

$$\mathbf{X}_{Y_i} = \mathbf{U}_{Y_i} \ \Sigma_{Y_i} \ \mathbf{W}_{Y_i} \tag{4.20}$$

where $\mathbf{U}_{Y_i} \in \mathbb{R}^{m \times m}$ is the left singular matrix containing the optimal POD bases of $\mathbf{Y}_i$ in its columns, Σ_{Y_i} is the diagonal matrix containing the singular values of the snapshot matrix $\mathbf{X}_{Y_i}$ in descending order. Now, the reduced representation of $\mathbf{Y}_i$ can be expressed as

$$\tilde{\mathbf{Y}}_i = {\mathbf{U}_{Y_i}^{q_i}}^{\top} \ \mathbf{Y}_i \tag{4.21}$$

where $\tilde{\mathbf{Y}}_i \in \mathbb{R}^{q_i}$ is the reduced representation of $\mathbf{Y}_i$, $\mathbf{U}_{Y_i}^{q_i} \in \mathcal{R}^{m \times q_i}$ is the orthonormal matrix containing q_i orthonormal basis vectors in its columns. Since, $\mathbf{Y}_i$ is computed from the difference between two consecutive levels of POD based approximations $\mathbf{v}_{i+1}$ and $\mathbf{v}_i$, i.e. $\mathbf{Y}_i = \mathbf{v}_{i+1} - \mathbf{v}_i$, the least–squares error in approximating $\mathbf{Y}_i$ by $(\mathbf{U}_{Y_i}^{q_i} \ \tilde{\mathbf{Y}}_i)$ is equivalent to the difference of two consecutive level ε (see Eq (4.19)) which is expressed as $\Delta\varepsilon_i = \varepsilon_i - \varepsilon_{i+1}$. Therefore, we expect $\mathbf{Y}_i$ to be attracted to a certain low dimensional subspace of dimension $q_i = \Delta r_i = (r_{i+1} - r_i)$ over all the levels. Next, we set $r_i = i$ and $q_i = \Delta r_i = (r_{i+1} - r_i) = q_c = 1$, and $q_c = 1$ for all $i = 0 \cdots I$.

Now the MLMC estimator (see Eq. (4.13)) for the expected value of $\mathbf{u}$ is expressed

as

$$\mathbb{E}[\mathbf{u}] = \sum_{i=0}^{I} \mathbb{E}[\mathbf{Y}_i] \approx \sum_{i=0}^{I} \mathbf{U}_{Y_i}^{q_i} \; \mathbb{E}[\tilde{\mathbf{Y}}_i] = \hat{\mathbf{u}}^{ml} = \sum_{i=0}^{I} \hat{\mathbf{Y}}_i = \sum_{i=0}^{I} \mathbf{U}_{Y_i}^{q_i} \; \hat{\tilde{\mathbf{Y}}}_i \tag{4.22}$$

In the fourth step, we extend the Multi-Level Multi-Fidelity method introduced in [59] by adopting multi-fidelity approach (see Eq. (4.24)) on every level of MLMC framework to derive an unbiased estimator of $\mathbb{E}[\tilde{\mathbf{Y}}_i]$ that takes the form

$$\hat{\tilde{\mathbf{Y}}}_i^{mf} = \hat{\tilde{\mathbf{Y}}}_i + \beta_i^1 \; (\hat{\tilde{\mathbf{Y}}}_i^1 - \hat{\tilde{\mathbf{Y}}}_i) + \sum_{f=2}^{F_i} \beta_i^f \; (\hat{\tilde{\mathbf{Y}}}_i^f - \hat{\tilde{\mathbf{Y}}}_i^{f-1}) \tag{4.23}$$

where $\tilde{\mathbf{Y}}_i^1 \cdots \tilde{\mathbf{Y}}_i^{F_i}$ are auxiliary random variables obtained from F_i number of level specific low-fidelity models of $\tilde{\mathbf{Y}}_i$, $\hat{\tilde{\mathbf{Y}}}_i^f$ estimates the expectation $\mathbb{E}[\tilde{\mathbf{Y}}_i^f]$ using M_i^f samples of $\tilde{\mathbf{Y}}_i^f$, $\beta_i^1 \cdots \beta_i^{F_i} \in \mathbb{R}$ are the coefficients on level i, $i = 0 \cdots I$. In this paper, we set $F_i = 1$ for all $i = 0 \cdots I$. Next, we use the optimization problem formulated in [113] to select optimal values M_{HF}^*, $M_i^{1^*}$ such that the mean square error of the MFML-MC estimator $\hat{\tilde{\mathbf{Y}}}_i^{mf}$ on every level is lower than the Monte Carlo estimator $\hat{\tilde{\mathbf{Y}}}_i$ for the same computational budget.

Now, the MFML-MC estimator for the expected value of $\mathbf{u}$ is obtained from replacing the expected values on the right hand side of Eq. (4.12) (also see Eq. (4.22)) by the ensemble averages

$$\hat{\mathbf{u}}^{ml} = \sum_{i=0}^{I} \hat{\mathbf{Y}}_i \approx \sum_{i=0}^{I} \mathbf{U}_{Y_i}^{q_i} \; \hat{\tilde{\mathbf{Y}}}_i^{mf} \tag{4.24}$$

In the fifth step, we utilize a data-driven approach to derive a level specific low-fidelity model $\tilde{\mathbf{Y}}_i^1$ in MFML-MC framework. In this data-driven approach, we first consider discrete nonlinear dynamical system on every level ($i = 0 \cdots I$) that takes the form

$$\tilde{\mathbf{Y}}_i^1(t+1) = \tilde{\mathbf{Y}}_i^1(t) + \mathbf{F}_i(\mathbf{x}, \tilde{\mathbf{Y}}_i^1(t)), \tag{4.25}$$

where $\mathbf{F}_i(\mathbf{x}, \tilde{\mathbf{Y}}_i^1(t))$ is the nonlinear term on level i for all $i = 0 \cdots I$ [104]. Next, we use GBTR [55] on every level to approximate $\mathbf{F}_i(\mathbf{x}, \tilde{\mathbf{Y}}_i^1(t))$. We use $(\mathbf{x}, \tilde{\mathbf{Y}}_i^1(t))$ as

an input to GBTR and compute $\mathbf{F}_i(\mathbf{x}, \tilde{\mathbf{Y}}_i^1(t))$ as an output. We fit GBTRs using the same training samples $\big((\mathbf{Y}_{i_1} \ \dots \ \mathbf{Y}_{i_T})^1 \ \dots \ (\mathbf{Y}_{i_1} \ \dots \ \mathbf{Y}_{i_T})^L\big)$ utilized in the third step. Finaly, we formulate the architecture of MF-DR-RNN [104] using the nonlinear residual equation defined as

$$\mathbf{r}_{t+1} = \mathbf{y}_{t+1} - \mathbf{y}_t - \mathbf{f}(\mathbf{x}, \mathbf{y}_t) \tag{4.26}$$

where $\mathbf{y} = [\tilde{\mathbf{Y}}_0 \cdots \tilde{\mathbf{Y}}_I]^\top \in \mathbb{R}^q$, $\mathbf{f}(\mathbf{x}, \mathbf{y}_t) = [\mathbf{F}_0(\mathbf{x}, \tilde{\mathbf{Y}}_0) \cdots \mathbf{F}_I(\mathbf{x}, \tilde{\mathbf{Y}}_I)]^\top \in \mathbb{R}^q$, $\mathbf{r}(t)$ is termed as the residual vector at time step t. Please note that, unlike model-based DR-RNN described in chapter 2 and chapter 3 [104] where the state variable at $t+1$ is utilized to compute $\mathbf{f}$, MF-DR-RNN utilize the state variable at t to compute $\mathbf{f}(\mathbf{x}, \mathbf{y}_t)$. Hence MF-DR-RNN [104] approximates the solution of Eq. (4.25) using the following iterative update equations

$$\begin{aligned} \mathbf{y}_{t+1}^{(k)} &= \mathbf{y}_{t+1}^{(k-1)} - \mathbf{w} \ \circ \ \phi_h(\mathbf{U} \ \mathbf{r}_{t+1}^{(k)}) \qquad && \text{for } k = 1, \\ \mathbf{y}_{t+1}^{(k)} &= \mathbf{y}_{t+1}^{(k-1)} - \frac{\eta_k}{\sqrt{G_k + \epsilon}} \ \mathbf{r}_{t+1}^{(k)} && \text{for } k > 1, \end{aligned} \tag{4.27}$$

where $\mathbf{U}, \mathbf{w}, \eta_k$ are the training parameters of MF-DR-RNN, ϕ_h is the tanh activation function, $\circ$ is an element-wise multiplication operator, $\mathbf{r}_{t+1}^{(k)}$ is the residual in layer k obtained by substituting $\mathbf{y}_{t+1} = \mathbf{y}_{t+1}^{(k-1)}$ into Eq. (4.26) and G_k is an exponentially decaying squared norm of the residual defined by

$$G_k = \gamma \ \|\mathbf{r}_{t+1}^{(k)}\|_2^2 + \zeta \ G_{k-1} \tag{4.28}$$

where γ, ζ are fraction factors and ϵ is a smoothing term to avoid divisions by zero [104]. The MF-DR-RNN output at each time step is defined as

$$\mathbf{y}_{t+1}^{(\text{RNN})} = \mathbf{y}_{t+1}^{K} \tag{4.29}$$

where K is the total number of layers in MF-DR-RNN architecture. The MF-DR-RNN parameters $\boldsymbol{\theta} = \{\mathbf{U}, \ \mathbf{w}, \ \eta_k\}$ are fitted by minimizing the mean square error defined

by

$$\mathbf{J}_{\text{MSE}}(\boldsymbol{\theta}) = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{t=1}^{T}(\mathbf{y}_t - \mathbf{y}_t^{(\text{RNN})})^2, \tag{4.30}$$

where $\mathbf{J}_{\text{MSE}}$ is the average distance between the reference solution $\mathbf{y}_t$ and the RNN output $\mathbf{y}_t^{\text{RNN}}$ across a number of samples L with time-dependent observations ($t = 1 \cdots T$ and $\ell = 1 \cdots L$) [107, 104]. The set of parameters $\boldsymbol{\theta}$ is estimated by Backpropagation Through Time (BPTT) [138, 122, 108, 100], Once the parameters $\boldsymbol{\theta}$ of MF-DR-RNN are fitted, we set $\tilde{\mathbf{Y}}_i^1 = \mathbf{y}_i^{\text{RNN}}$ (see Eq. (4.23)) for all the levels $i = 0 \cdots I$.

4.5 Numerical Experiments

In this section, we present numerical results to evaluate the performance of MFML-MC method. The numerical results are based on two UQ tasks involving two-phase flow in heterogeneous porous media domain. The two test cases are quarter five spot problem and the uniform flow problem with the uncertainties in the permeability field [84]. In subsection 4.5.1, we describe high-fidelity model setup, in subsection 4.5.2, we describe low-fidelity model setup in order to formulate MFML-MC framework, and in subsection 4.5.3, we define a set of error metrics that we utilize to compare MFML-MC method with standard MC method that uses either high-fidelity model or low-fidelity model. In subsection 4.5.4, we provide the results for quarter five spot problem and in subsection 4.5.5, we provide the results for uniform flow problem.

4.5.1 High-fidelity model setup

We consider two-phase flow of oil and water in a two dimensional porous media domain $[0\ 1] \times [0\ 1]$ where water is injected to displace the residual oil. We consider Eq. (4.3) as a high-fidelity model to describe the flow behaviour of oil and water. We define the relative permeability based on Corey's model $k_{rw}(s) = {s^*}^2$, $k_{ro} = (1-s^*)^2$, where $s^* = (s - s_{wc})/(1 - s_{or} - s_{wc})$, s_{wc} is the irreducible water saturation and s_{or}

is the residual oil saturation [1]. We set $s_{wc} = 0.2$, $s_{or} = 0.2$, and initial water saturation to $s_{wc}(0.2)$. We set the porosity field in the porous media domain to a constant value of 0.2. We set viscosity ratio of water and oil to 0.2. We consider uncertainties from the permeability field and assumed to be modelled as a log-normal distribution function with zero mean and exponential covariance that takes the form

$$\mathbb{C}ov = \sigma_k \ \exp\left[-\frac{|x_1 - x_2|}{\iota_k}\right] \tag{4.31}$$

where σ_k is the variance, ι_k is the correlation length, x_1 and x_2 are the points in the grid domain. We set σ_k to 1 and ι_k to 0.1. Sample realisations of log-permeability values are displayed in Figure 4.2.

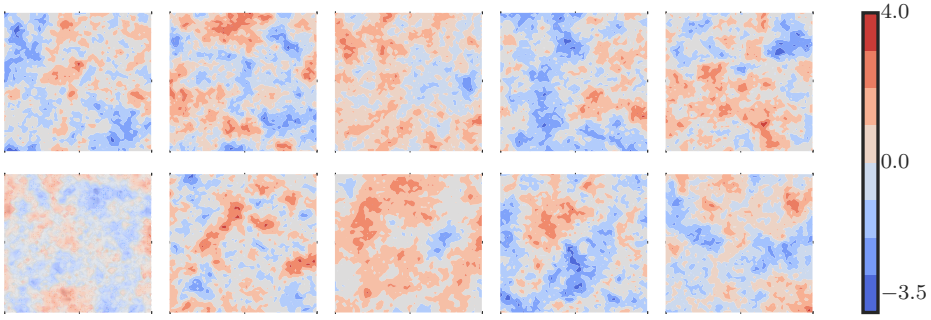


Figure 4.2: Sample plots of log-permeability field. Uncertain permeability field is modelled from a log-normal distribution function with zero mean and exponential covariance.

As mentioned in section 4.2, we use sequential formulation to solve Eq. (4.3) for pressure and water saturation [1]. We first generate an uniform mesh of 96×96 blocks in a spatial domain. We use finite volume method with two point flux approximation [1] to solve for pressure and an upwind finite-volume method with an implicit backward Euler method combined with Newton-Raphson iterative method to solve for saturation. We set time step size to 0.015 and we solve Eq. (4.3) for 200 time steps. We solve pressure and update velocity field at every 8th time step as pressure field changes much slower than saturation field over the time. The unit of time a non dimensional unit called pore volumes injected (PVI) [78]. The pictorial representation of the algorithm utilized in high-fidelity model setup is displayed in Figure (chapter 3). This sequential formulation algorithm is coded in Python programing langauage to generate high-fidelity solution data and the code is attached

in the appendix.

As defined in section 4.2, QoI is $\mathbf{u} \in \mathbb{R}^m$, where $u_i = y_s(x_i, y_i)$, $i = 1 \cdots m \ll n$ at specific time steps. The first moment estimate of $\mathbf{u}(t)$ at specific time steps are the desired statistic. The interested grid points $((x_i, y_i) \;\; i = 1 \cdots m)$ are 6×6 grid points ($m = 36$) uniformly selected from the 96×96 spatial domain. The interested time steps are $t = 10, 20, \cdots, 200$. We solve Eq. (4.3) for 25000 random realisations of the permeability field and use Monte Carlo method to estimate the statistics of $\mathbf{u}$ [78].

4.5.2 Low-fidelity model setup

We first compute the optimal POD bases matrices $\mathbf{U}_u$ and $\mathbf{U}_{Y_i}$ for all $i = 0 \cdots I$. We compute the POD matrices from the SVD of the snapshot matrices $\mathbf{X}_u, \mathbf{X}_{Y_i}, i = 0 \cdots I$. We built the snapshot matrices from 10 random samples of high-fidelity model solution data. In order to select the 10 random high-fidelity models to build snapshot matrices, we use clustering algorithm to cluster 25000 random permeability realizations into 10 clusters [61]. Then, we solve the high-fidelity model for a single permeability realization from each cluster.

Following that, the obtained matrix $\mathbf{U}_u$ is utilized to build a sequence of POD approximations of $\mathbf{u}$ (as detailed in the section 4.4) from the collected training data. Then the matrix $\mathbf{U}_{Y_i}^{q_i}$ is utilized to compute training samples of $\tilde{\mathbf{Y}}_i$ (the reduced representation of $\mathbf{Y}_i$) for all $i = 0 \cdots I$. We set $I = 18$, $r_i = i$, and therefore $q_i = 1$ as already mentioned in the section 4.4.

Next, to derive (MF-DR-RNN), we first build GBTR on every level ($i = 0 \cdots I$) to estimate $\mathbf{F}_i$ using Scikit-learn [112] a machine learning python package to implement the GBTRs. We use the training samples of $\tilde{\mathbf{Y}}_i$ to fit the level specific GBTR. and to obtain the training samples of $\mathbf{y}_{t+1}$, $\mathbf{y}_t$, and $\mathbf{f}(\mathbf{x}, \mathbf{y}_t)$. We utilize PyTorch framework [111] (a deep learning python package with Torch library as a backend) to implement the MF-DR-RNN. In all the numerical test cases, we utilize MF-DR-RNN with six layers ($K = 6$ in Eq. (4.27)). Further, we optimize the MF-DR-RNN model parameters using rmsprop algorithm [130, 111] as implemented

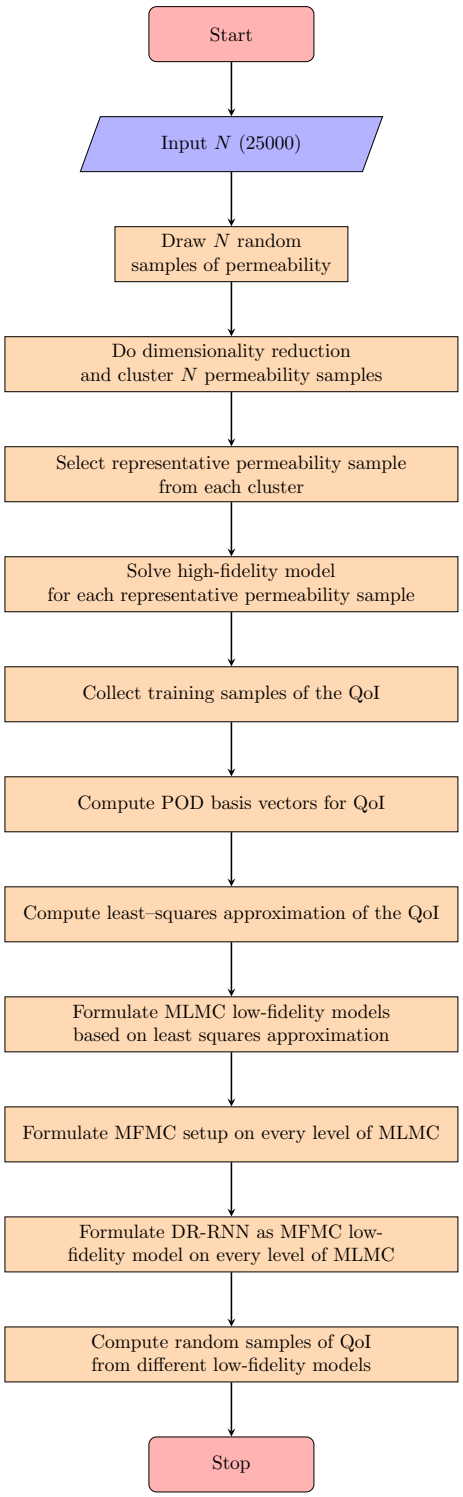


Figure 4.3: Flowchart describing the low-fidelity model setup (section 4.5.2).

in PyTorch, where we set the weighted average parameter to 0.9 and the learning rate to 0.001. The weight matrix $\mathbf{U}$ in Eq. (4.27) is initialized randomly from the uniform distribution function $\mathtt{U}[0.01, 0.02]$. The vector training parameter $\mathbf{w}$ in Eq. (4.27) is initialized randomly from the uniform distribution function $\mathtt{U}[0.1, 0.5]$. The scalar training parameters η_k in Eq. (4.27) are initialized randomly from the

uniform distribution $\mathtt{U}[0.1, 0.4]$. We set the hyperparameters ζ and γ in Eq. (4.28) to 0.9 and 0.1, respectively. The MF-DR-RNN is trained to approximate $\mathbf{y}_{t+1}$. The parameters of the MF-DR-RNN are trained by minimizing the loss function defined in Eq. (4.30) using the training samples of $\mathbf{y}_{t+1}$. Figure 4.3 displays the flowchart denoting the outline of the low-fidelity model setup.

4.5.3 Evaluation metrics

We evaluate the performance of MFML-MC method using two time specific error metrics defined by

$$\begin{aligned} \hat{e}_t^{\text{bias}} &= \frac{1}{N_{\text{e}}} \sum_{j=1}^{N_{\text{e}}} \|\hat{\mathbf{u}}_t^{\text{ref}} - \hat{\mathbf{u}}_t^{(j)}\|_2^2 \\ \hat{e}_t^{\epsilon} &= \frac{1}{N_{\text{e}}} \sum_{j=1}^{N_{\text{e}}} \mathbb{V}\text{ar}(\hat{\mathbf{u}}_t^{(j)}) \end{aligned} \tag{4.32}$$

where N_{e} is the number of runs utilized to estimate the errors, $\hat{\mathbf{u}}_t^{\text{ref}}$ is the reference result of $\mathbb{E}[\mathbf{u}_t]$ obtained from Monte Carlo estimate $\hat{\mathbf{u}}_t^{(\text{MC})}$ computed with $N = 25000$ high-fidelity model samples. $\hat{\mathbf{u}}_t^{(j)}$ is the approximation of $\mathbb{E}[\mathbf{u}_t]$ that can be obtained from various estimators including Monte Carlo estimate that uses only high-fidelity model, Monte Carlo estimate that uses only low-fidelity model, and the MFML-MC estimator. $\hat{\mathbf{u}}_t^{(j)}$ is obtained for a fixed computational budget p. $\hat{\mathbf{u}}_t^{(j)}$ is evaluated from different independent samples for each $j = 1 \cdots N_{\text{e}}$.

Additionally, we utilize two global error metrics defined as

$$\begin{aligned} \hat{e}^{\text{bias}} &= \frac{1}{N_{\text{e}}} \sum_{j=1}^{N_{\text{e}}} \max_{t=1}^{T} \|\hat{\mathbf{u}}_t^{\text{ref}} - \hat{\mathbf{u}}_t^{(j)}\|_2^2 \\ \hat{e}^{\epsilon} &= \frac{1}{N_{\text{e}}} \sum_{j=1}^{N_{\text{e}}} \max_{t=1}^{T} \mathbb{V}\text{ar}(\hat{\mathbf{u}}_t^{(j)}) \end{aligned} \tag{4.33}$$

where all the time snapshots of $\mathbf{u}$ are used. We set $N_{\text{e}} = 15$ to evaluate the two time specific error metrics and the two global error metrics.

4.5.4 Numerical test case 1

Test case 1 is two dimensional quart-five spot problem where water is injected in the lower left corner $(0, 0)$ of the porous media domain to produce oil and water in the top right corner $(1, 1)$ [84]. We set q defined in the saturation equation Eq. (4.3) to 0.05 at $(0, 0)$ and -0.05 at $(1, 1)$. We set no flux boundary condition in all the four sides of the porous media domain. The left panel of Figure 4.4 displays the quart-five spot problem set up and the right panel of Figure 4.4 displays the decay of the singular values of the snapshot matrix $\mathbf{X}_u$.

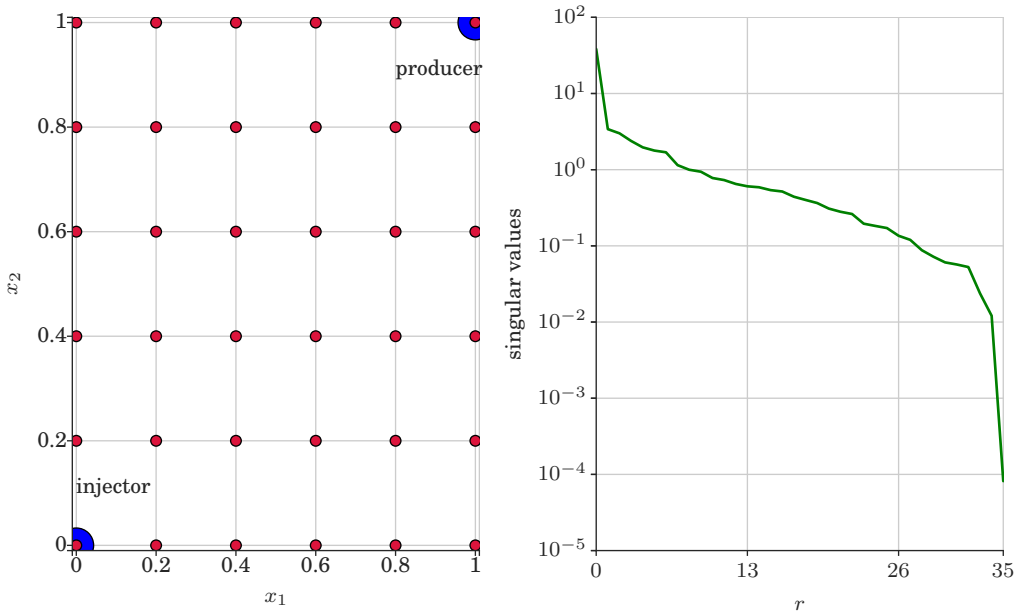


Figure 4.4: Test case 1. Left: Two dimensional quart-five spot problem set-up where water is injected in the lower left corner (blue dot). Oil is displaced and produced with water in the upper right corner (blue dot). The red dots denotes spatial locations in the porous media domain where the statistics of the QoI $\mathbf{u}$ are investigated. Right: Decay of singular values of the snapshot matrix $\mathbf{X}_u$.

Figure 4.5 shows the results for the estimation of $\mathbb{E}[\mathbf{u}_t]$ (first moment of $\mathbf{u}$) obtained from the reference result (MC estimate with 25000 samples) and from various MC estimators. In Figure 4.5, MC estimator that uses only high-fidelity model is denoted as MC-HF and that uses only low-fidelity model is denoted as MC-LF. In Figure 4.5, results shown in the top row are obtained at time = 0.3 PVI and results shown in the bottom row are obtained at time = 0.8 PVI. As shown in Figure 4.5, the estimation of $\mathbb{E}[\mathbf{u}_t]$ obtained from MC-LF deviates significantly from the reference result. This clearly shows that utilizing only low-fidelity model

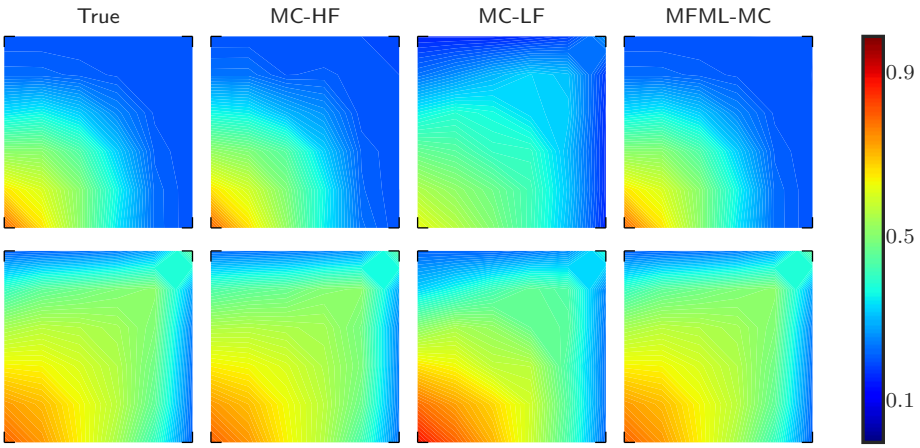


Figure 4.5: Test case 1: Comparison of estimation of $\mathbb{E}[\mathbf{u}_t]$ (mean water saturation field at 6×6 spatial grid) for a fixed computational budget $p = 100$, where p is the number of MC realizations that uses only high-fidelity model . Top Row: Estimation of $\mathbb{E}[\mathbf{u}_t]$ at time $t = 0.3$ PVI. Bottom Row: Estimation of $\mathbb{E}[\mathbf{u}_t]$ at time $t = 0.8$ PVI.

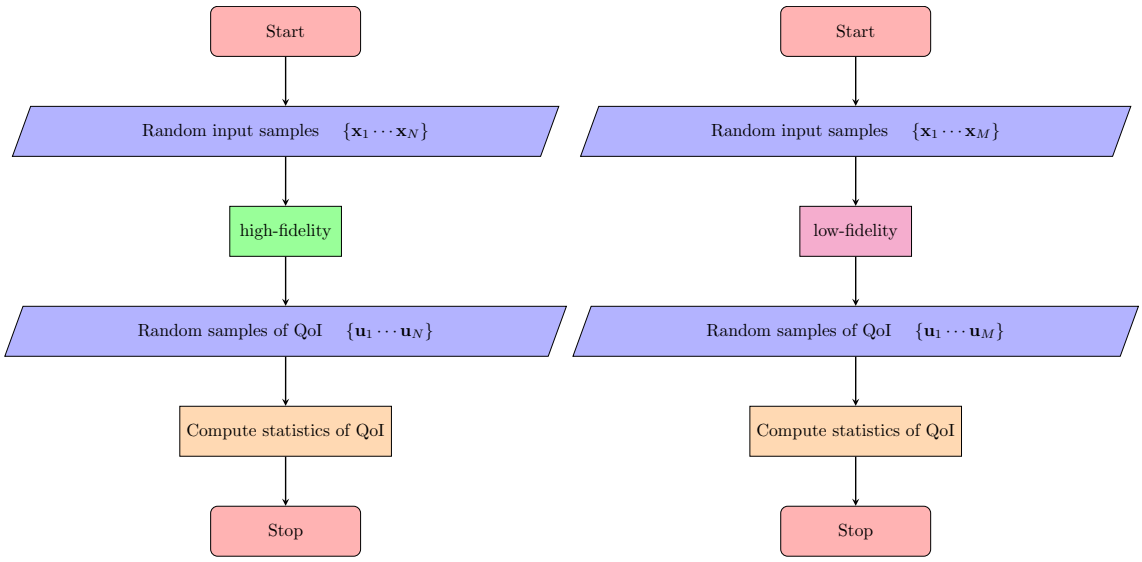


Figure 4.6: Flowchart diagram describing the standard Monte-Carlo method to estimate the statistics of QoI. Left: standard Monte-Carlo method that uses only high-fidelity model (MC-HF method). Right: standard Monte-Carlo method that uses only only low-fidelity model (MC-LF method).

in MC framework resultant in biased estimation with respect to the reference result. Furthermore, Figure 4.5 shows that the estimation of $\mathbb{E}[\mathbf{u}_t]$ obtained from MFML-MC estimator is almost indistinguishable from the reference result. This result confirm that combining higher number of low-fidelity model realizations with the high-fidelity model in MFML-MC framework can improve the estimator of the first moment of the saturation field.

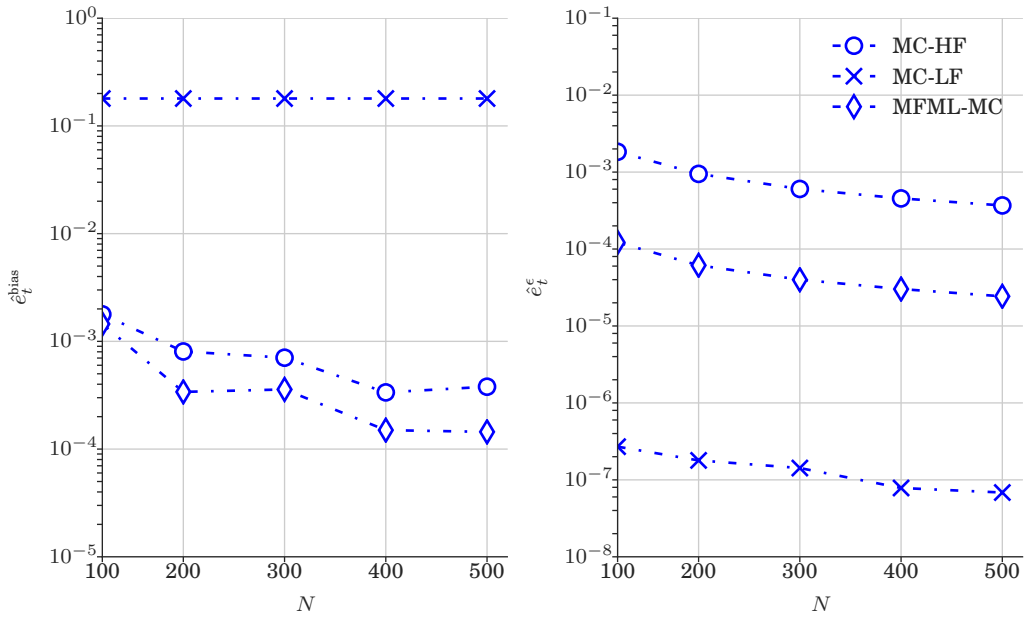


Figure 4.7: Test case 1: Plot of $\hat{e}_t^{\text{bias}}$, and $\hat{e}_t^{\epsilon}$ (Eq. (4.32)) for the estimation of $\mathbb{E}[\mathbf{u}_t]$ (water saturation field at 6×6 spatial grid) obtained from various estimators. $\hat{e}_t^{\text{bias}}$ and $\hat{e}_t^{\epsilon}$ are shown as a function of computational budget $p = [1, 2, 3, 4, 5] \times 10^2$, where p is the number of MC realizations that uses only high-fidelity model. Left: $\hat{e}_t^{\text{bias}}$ at time $t = 0.3$ PVI. Right: $\hat{e}_t^{\epsilon}$ at time $t = 0.3$ PVI.

Figure 4.7 reports the comparison of $\hat{e}_t^{\text{bias}}$ and $\hat{e}_t^{\epsilon}$ (see Eq. (4.32)) obtained from various estimators. The left of Figure 4.7 reports $\hat{e}_t^{\text{bias}}$ and the right of Figure 4.7 reports $\hat{e}_t^{\epsilon}$ as a function of computational budget $p = [1, 2, 3, 4, 5] \times 10^2$, where p is the number of MC-HF realizations. The results of $\hat{e}_t^{\text{bias}}$ from Figure 4.7 shows that Monte Carlo estimator that uses MC-LF is a biased estimator of the mean QoI value. The results of MFML-MC estimator displayed in left of Figure 4.7 confirm that the MFML-MC estimator is an unbiased estimator of the expectation. This shows that despite the low-fidelity model is a poor approximation of the high-fidelity model, the error of the MFML-MC estimator can be significantly reduced if the low-fidelity model is combined with the high-fidelity. The right of Figure 4.7 shows that the variance of the MFML-MC and MC-LF estimators are at least an order of magnitude less when compared to MC-HF. Nevertheless, while MC-LF is a biased estimator as shown in left of Figure 4.7, MFML-MC estimator that uses the low-fidelity model in combination with the high-fidelity model is an unbiased estimator of the expectation.

Figure 4.8 reports the comparison of $\hat{e}^{\text{bias}}$ and $\hat{e}^{\epsilon}$ (see Eq. (4.33)) obtained from various estimators. We can clearly observe the trend of $\hat{e}^{\text{bias}}$, and $\hat{e}^{\epsilon}$ in Figure 4.8 are

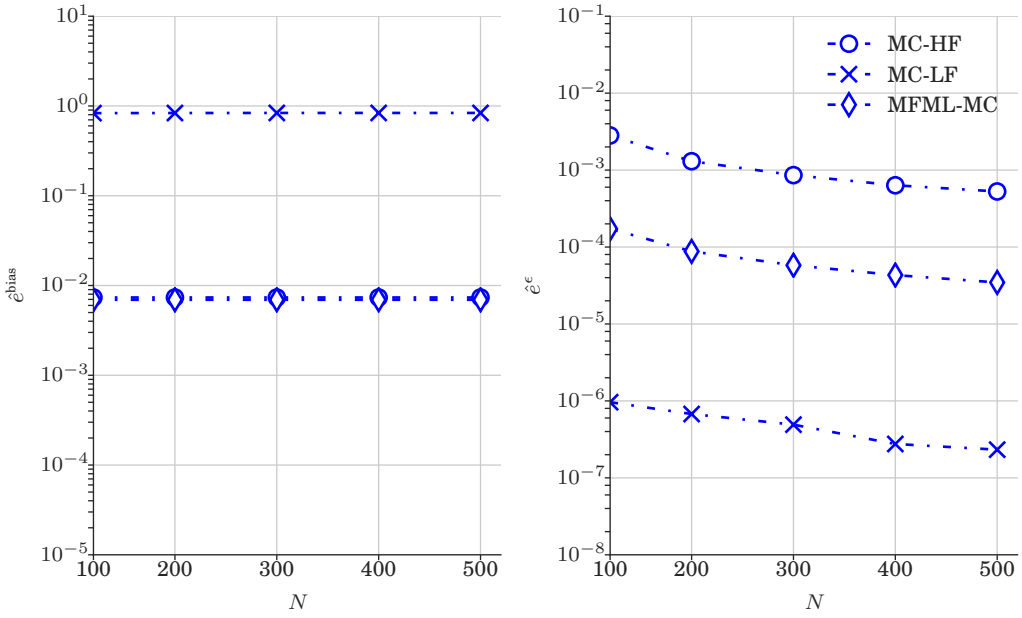


Figure 4.8: Test case 1: Plot of $\hat{e}^{\text{bias}}$ and $\hat{e}^{\epsilon}$ (Eq. (4.33)) estimation of $\mathbb{E}[\mathbf{u}]$ (water saturation field at 6×6 spatial grid) obtained from various estimators. $\hat{e}^{\text{bias}}$ and $\hat{e}^{\epsilon}$ are shown as a function of computational budget $p = [1, 2, 3, 4, 5] \times 10^2$, where p is the number of MC realizations that uses only high-fidelity model.

similar to the one observed in Figure 4.7 which confirms that MFML-MC method leads to variance reduction with unbiased estimation at all time steps.

Table 4.1 compare the speedup factors of MFML-MC method with respect to the Monte Carlo estimator that uses the high-fidelity model only. In Table 4.1, MFML-MC achieves a speedups with respect to MC-HF that range from 8 up to 15 for the same specific ϵ.

Table 4.1: Performance chart of MFML-MC estimator for test case 1. ϵ defined in Eq. (4.7) is shown as a function of computational budget p, where p is the number of MC realizations that uses the high-fidelity model only. ϵ is estimated at time = 0.3 PVI.

ϵ	p	CPU Time (min)		Speedup
		MC-HF	MFML-MC	
10^{-4}	5×10^2	125	15	8.3
10^{-5}	9×10^3	2250	210	10.5
10^{-6}	25×10^3	6250	490	13.4

4.5.5 Numerical test case 2

Test case 2 is a two dimensional uniform flow problem where water is injected from the left side of the porous media domain to produce oil and water from the right side. We set no flow boundary conditions in the remaining two sides (top and bottom) of the domain. We set inflow rate to 0.08 and outflow rate to 0.08 due to incompressibility constraint set in the problem [84]. The left panel of Figure 4.9 displays uniform flow problem set up and the right panel of Figure 4.9 displays the decay of the singular values of the snapshot matrix $\mathbf{X}_u$.

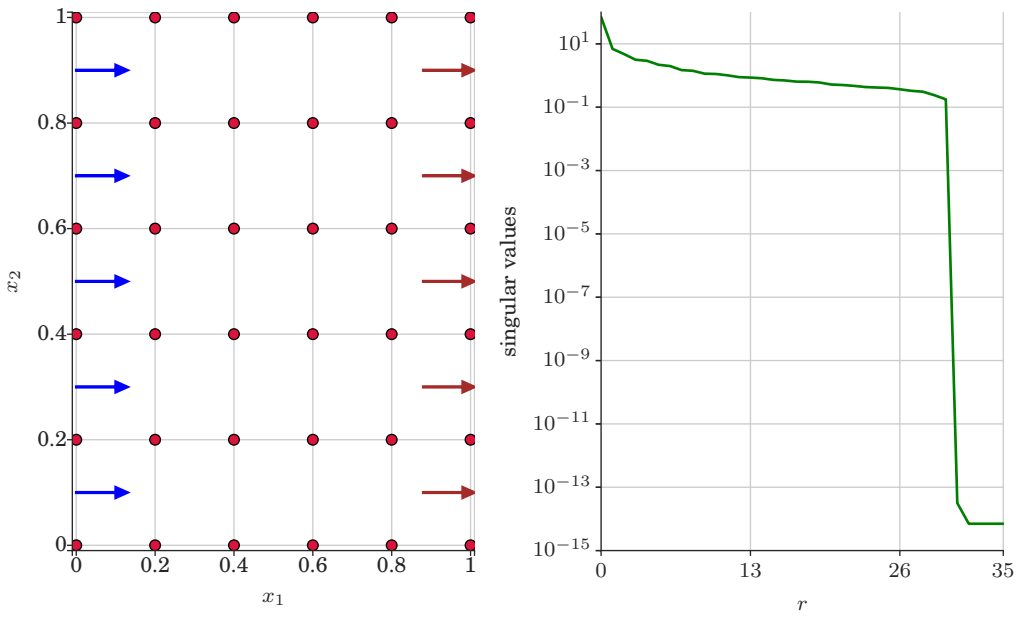


Figure 4.9: Test case 2 Left: Uniform flow problem set up where water is injected from the left side denoted by blue arrows. Oil and water are produced from the right side denoted by brown arrows. The red dots denotes the spatial locations of the porous media domain where the statistics of the QoI $\mathbf{u}$ are investigated. Right: Decay of singular values of the snapshot matrix $\mathbf{X}_u$.

Figure 4.10 show the results for the first moment of the saturation field ($\mathbf{u}$) obtained from the reference result (MC estimate with 25000 samples) and from various MC estimators. The display settings defined in Figure 4.10 are the same to the one defined in Figure 4.5. In Figure 4.10, we can see that the results obtained from MFML-MC method is almost indistinguishable from the reference results whereas MC-LF yields extremely inaccurate results.

Figure 4.11 reports the comparison of $\hat{e}_t^{\text{bias}}$ and $\hat{e}_t^{\epsilon}$ (see Eq. (4.32)) obtained from various estimators. The variance reduction can be clearly observed in Figure 4.11

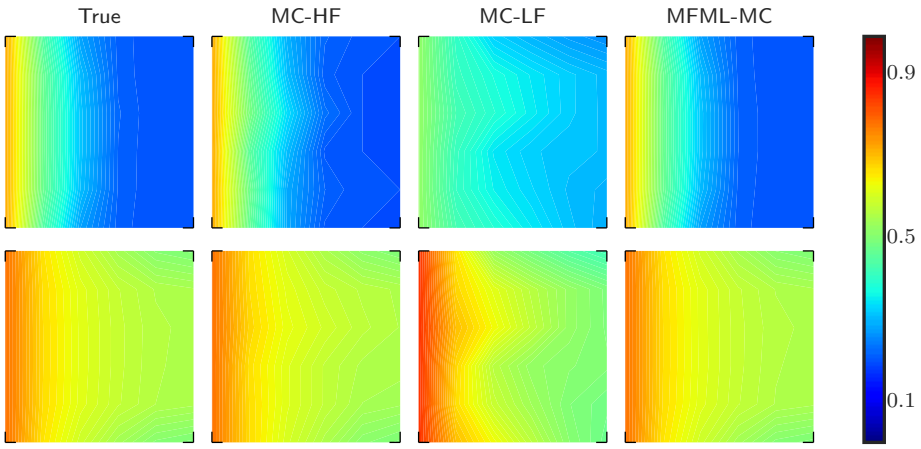


Figure 4.10: Test case 2: Comparison of estimation of $\mathbb{E}[\mathbf{u}_t]$ (mean water saturation field at 6×6 spatial grid) for a fixed computational budget $p = 100$, where p is the number of MC realizations that uses only high-fidelity model . Top Row: Estimation of $\mathbb{E}[\mathbf{u}_t]$ at time $t = 0.3$ PVI. Bottom Row: Estimation of $\mathbb{E}[\mathbf{u}_t]$ at time $t = 0.8$ PVI.

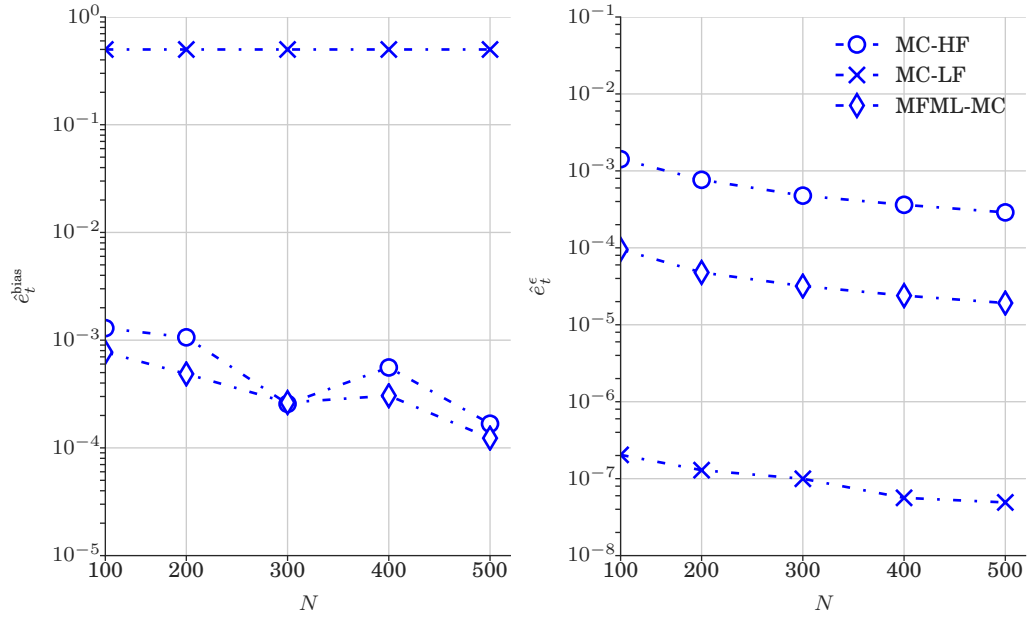


Figure 4.11: Test case 2: Plot of $\hat{e}_t^{\text{bias}}$ and $\hat{e}_t^{\epsilon}$ (Eq. (4.32)) for the estimation of $\mathbb{E}[\mathbf{u}_t]$ (water saturation field at 6×6 spatial grid) obtained from various estimators. $\hat{e}_t^{\text{bias}}$ and $\hat{e}_t^{\epsilon}$ are shown as a function of computational budget $p = [1, 2, 3, 4, 5] \times 10^2$, where p is the number of MC realizations that uses only high-fidelity model. Left: $\hat{e}_t^{\text{bias}}$ at time $t = 0.3$ PVI. Right: $\hat{e}_t^{\epsilon}$ at time $t = 0.3$ PVI.

and the trend of Figure 4.11 is similar to the one observed in Figure 4.7 (Test case 1). The results of Figure 4.11 again confirm that combining the high-fidelity model with the low-fidelity model leads to a variance reduction. Please note that a similar confirmation was observed in Figure 4.7.

Figure 4.12 reports the comparison of $\hat{e}^{\text{bias}}$ and $\hat{e}^{\epsilon}$ (see Eq. (4.33)) obtained from

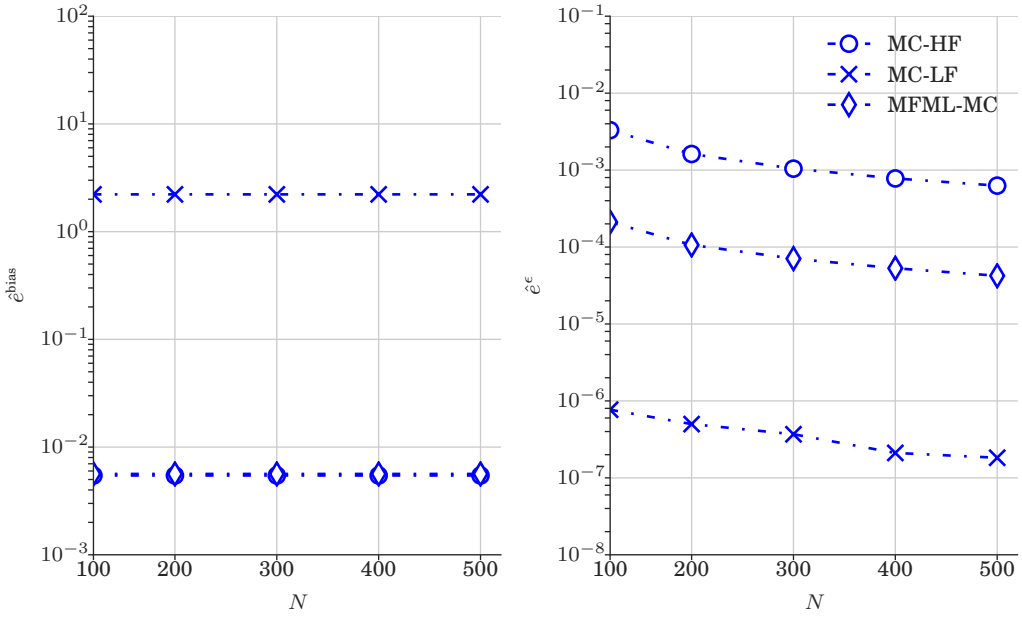


Figure 4.12: Test case 2: Plot of $\hat{e}^{\text{bias}}$ and $\hat{e}^{\epsilon}$ (Eq. (4.33)) for the estimation of $\mathbb{E}[\mathbf{u}]$ (water saturation field at 6×6 spatial grid) obtained from various estimators. $\hat{e}^{\text{bias}}$ and $\hat{e}^{\epsilon}$ are shown as a function of computational budget $p = [1, 2, 3, 4, 5] \times 10^2$, where p is the number of MC realizations that uses only high-fidelity model.

various estimators. As observed in Figure 4.8, the results displayed in Figure 4.11 shows that MFML-MC method leads to variance reduction with unbiased estimation.

Table 4.2 compare the speedup factors of MFML-MC method with respect to the MC method that uses the high-fidelity model only. In Table 4.2, MFML-MC achieves a speedups with respect to MC-HF that range from 10 up to 19 at a specific ϵ.

Table 4.2: Performance chart of MFML-MC estimator for test case 2. ϵ defined in Eq. (4.7) is shown as a function of computational budget p, where p is the number of MC realizations that uses the high-fidelity model only. ϵ is estimated at time = 0.3 PVI.

ϵ	p	CPU Time (min)		Speedup
		MC-HF	MFML-MC	
10^{-4}	5×10^2	148	14	10.8
10^{-5}	9×10^3	2850	197	14.5
10^{-6}	25×10^3	7950	410	19.4

4.6 Conclusion

In this chapter, we proposed a MFML-MC method combining the features of both the MFMC method and the MLMC method. In MFML-MC method, we formulated MLMC framework with a sequence of POD approximations of high-fidelity model outputs. Furthermore, in MFML-MC method, we formulated a MFMC setup on every level of MLMC framework in order to compute an unbiased statistical estimation. Finally, we utilized GBTR in the MFMC setup to formulate a level specific low-fidelity model.

We applied MFML-MC method on two uncertainty quantification problems involving two-phase flows in random heterogeneous porous media. The uncertain permeability field is modeled from log-normal distribution function with exponential covariance function. Estimate of the first statistical moments of the water saturation at uniformly selected spatial grid points over a specific instants in time are calculated by MFML-MC, MC-HF, and MC-LF methods. Comparisons between MFML-MC and MC-LF showed that MC-LF as a biased estimator and MFML-MC estimator as an unbiased estimator of the expectation. Comparisons between the MFML-MC and MC-HF computing times showed speedups of MFML-MC with respect to MC-HF that ranged from 8 up to 19 at equivalent accuracy.

Future work should consider the extension of MFML-MC method by utilizing two or more level specific low-fidelity models in the MFMC setup. In addition, it will also be interest to use MFML-MC method for history matching [48, 49], where we aim to minimize the mismatch between field observation data and the one computed from the high-fidelity model simulations by adjusting the geological model parameters. Future work should also verify the applicability of MFML-MC method for large-scale realistic problems with many wells and time varying injection rates by which the potential of MFML-MC method in speeding up a realistic Monte Carlo simulation can be magnified.

Chapter 5

Conclusions

In this thesis, we addressed the need for reducing the computational cost of Monte Carlo method that uses high-fidelity nonlinear dynamical systems. In particular, we focused on nonlinear dynamical systems representing the multi-phase flow through the subsurface porous media. Multi-phase flow through multi-scale subsurface materials is a complex process and simulating such processes are typically a computationally intensive task owing to the large systems of nonlinear equations that must be solved at each time step. Moreover, forward propagation of uncertainties through those complex systems requires a large number of simulations, and thus is usually computationally prohibitive in a realistic scenario. To address such challenges in the complex process, we developed low-fidelity models based on Deep Learning techniques, and an appropriate Monte Carlo framework based on multi-fidelity techniques. In this chapter, we summarize the research topics studied in this dissertation, and highlight the suggestions for continuing work on this topic.

5.1 Thesis Summary

In Chapter 2, we addressed the issue related to the cost of solving large systems of nonlinear equations at each time step. More specifically, we proposed a physics aware RNN architecture namely DR-RNN in order to reduce the computational complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ for fully coupled nonlinear systems of size n at each time step. In the numerical results for three UQ tasks, we showed that the

PDF obtained from DR-RNN with residual layers closely followed the trend of the reference PDF. We also showed that DR-RNN had much lower model complexity than standard RNN models and hence had much lower computing times during training the parameters. We further combined DR-RNN with POD-Galerkin projection based ROM for further reduction of the computational complexity. The idea of POD-Galerkin projection technique is to projection the mathematical equations of the dynamical system onto the space generated by a low number of basis functions. POD basis are computed by applying singular value decomposition of the training snapshot solution matrix. We applied our approaches to two synthetic UQ problems. In the unsteady heat diffusion UQ problem, our approach yielded the same accuracy as POD based ROM but with a less computational complexity. In the fluid displacement problem within an one dimensional porous media, we employed DEIM with POD to approximate the nonlinear term so that the resulting reduced system is independent of the full-order model dimension. In both POD-DEIM ROM and DR-RNN that approximated the POD-DEIM ROM, we observed an increase in the accuracy in approximating the FOM as we increased the number of POD basis. However, DR-RNN constructed an accurate reduced system without evaluating the Jacobian matrix and thus had less computational complexity in comparison to the POD-DEIM ROM.

Assessing the performance of DR-RNN in the two and three dimensional porous media flow problems is a crucial point to demonstrate the potential of DR-RNN in speeding up a realistic Monte Carlo simulation. Thus, in Chapter 3, we focused our attention on implementing a reduced order model using DR-RNN for the solution of quarter five spot and uniform flow problems, where uncertain permeability field was modeled as a log-normal distribution function. More precisely, we developed two POD-Galerkin based ROM using DR-RNN architecture namely, DR-RNN$^{\mathrm{p}}$ (DR-RNN combined with POD-Galerkin) and DR-RNN$^{\mathrm{pd}}$ (DR-RNN combined with POD-Galerkin and DEIM). We provided a set of error metrics to evaluate the performance of the DR-RNN$^{\mathrm{p}}$ and the DR-RNN$^{\mathrm{pd}}$. The errors are evaluated by comparing the full model and the reduced model solutions. In the numerical test cases,

DR-RNN$^{\text{p}}$ and the DR-RNN$^{\text{pd}}$ not only gave an accurate reduced system that is substantially smaller than the original system with a general nonlinearity, but it also preserved the stable behaviour of the original system. Specifically, comparison of mean water saturation field over the simulation time showed that DR-DR-RNN$^{\text{p}}$ and DRRNN$^{\text{pd}}$ results were very close to the least–square solutions while POD-Galerkin ROM yielded extremely inaccurate and unstable results. Furthermore, we observed that errors obtained from DR-RNN$^{\text{p}}$ and DR-RNN$^{\text{pd}}$ decreased when we increased the number of POD basis whereas increasing the number of basis vectors in POD-Galerkin ROM did not lead to improved accuracy. The numerical results also illustrated the possibility of constructing a single POD basis that can be used for UQ tasks with a very large number of uncertain parameters.

In Chapter 2 and 3, we replaced the high-fidelity model with a single low-fidelity model and estimated the statistics of the high-fidelity model outputs at reduced computational cost. The key to the effectiveness of utilizing single low-fidelity model is that the absolute errors of the low-fidelity model outputs with respect to the high-fidelity model outputs should be within the error bounds. Failing to satisfy such constraints in the error bound could result in an inherent biased statistical estimations. As a consequence, in many engineering applications, there is a hard restriction to utilize low-fidelity model although a suite of models is available to estimate the statistics. Thus, in Chapter 4 we focused our attention on rigorous multi-fidelity approaches to derive an unbiased estimator by combining an arbitrary number of low-fidelity models with the high-fidelity model. More precisely, we proposed a multi-fidelity approach and derived MFML-MC estimator by combining the features of both the MFMC method and the MLMC method. MF-DR-RNN, a variant of DR-RNN was derived and utilized as a level specific low-fidelity model in order to guarantee unbiased estimation and the variance reduction. The performance of the MFML-MC estimator was evaluated on forward propagation of uncertainties subject to uncertain permeability field. The QoI are the water saturation values at specific grid points over the specific time steps. In the numerical results, we demonstrated MFML-MC estimator to be superior in comparison to MC-HF and MC-LF

estimators. More precisely, the results for mean saturation field confirmed that combining higher number of low-fidelity model realizations with the high-fidelity model in MFML-MC framework leads to an unbiased estimation. In addition to unbiased estimation, MFML-MC framework also enabled variance reduction at least an order of magnitude less when compared to MC-HF for a fixed computational budget. As a consequence, MFML-MC framework showed speedups of MLMC with respect to MC-HF that range from 8 up to 19 at equivalent accuracy. Moreover, MFML-MC speedups could scale well with increasing the number of grid blocks (n on the order of 10^6 to 10^8) in the the high-fidelity model. Thus, MFML-MC framework would be suitable for problems with a very large number of uncertain parameters in large-scale industrial applications.

5.2 Future Work

- DR-RNN techniques presented in this thesis were only applied to two phase incompressible flow without gravity, and capillary pressure forces. In the future, one should investigate and should extend DR-RNN techniques to handle three phase flow with multiple components.

- The numerical test cases in this thesis only considered Gaussian permeability fields. The application of DR-RNN techniques should be extended to handle non-Gaussian permeability fields modeled with multi-point simulation techniques.

- The numerical test cases in this thesis involved only few wells with constant injection rate. Thus future work should be verified for large-scale realistic problems containing grid blocks on the order of 10^6 to 10^8 grid blocks with many wells and time varying injection rates.

- In this thesis, DR-RNN techniques were applied to systems over a wide range of input parameters. Thus, the use of DR-RNN techniques in history-matching procedure is also a promising direction for future work. It should also be possible to extend DR-RNN techniques for optimization under geological uncer-

tainly. Such extension would be able to provide effective closed loop reservoir management and reliable forecasts for cases with different well controls and different geological parameters.

- In this thesis, we formulated DR-RNN techniques where we combined DR-RNN with proper orthogonal decomposition and discrete empirical interpolation method. It may also be possible to combine DR-RNN with Upscaling technique.

- In this thesis, we presented MFML-MC method where we utilized Model-Free DR-RNN as a level specific low-fidelity model. An apparent extension should be utilizing two or more level specific low-fidelity models.

- Other possibility of extending MFML-MC method should be considered. For example, MFML-MC method can be formulated on every level of the standard MLMC method which is constructed from a hierarchy of coarse spatial and/or time discretization models.

- In situations where high levels of nonlinearity exist like in compositional reservoir simulation, linear dimensionality reduction from proper orthogonal decomposition may degrade the accuracy of the results. In the future, it will be useful to develop more sophisticated nonlinear dimensionality reduction techniques in addressing this issue.

Appendix A

```
# in the name of almighty
# Assalamu Alaikum
from __future__ import division
import numpy as np
import matplotlib.pyplot as plt
import scipy
import scipy.io
import cPickle as pickle
import prettyplotlib as pplt


class Params(object):
    def __init__(self):
        '''
        input parameters defined here and used in
        pressure and saturation classes
        '''
        print 'salam preparing input params ... '
        Grid = {}
        Dx = 1.  # x direction domain length
        Dy = 1.  # y direction domain length
        Dz = 1.  # z direction domain length
        Grid['Nx'] = 64  # number of points in x direction
        Grid['Ny'] = 64  # number of points in y direction
        Grid['Nz'] = 1   # number of points in z direction
        Grid['hx'] = Dx / Grid['Nx']
        Grid['hy'] = Dy / Grid['Ny']
```

```
        Grid['hz'] = Dz / Grid['Nz']
        # number of points
        N = Grid['Nx'] * Grid['Ny'] * Grid['Nz']
        Grid['V'] = Grid['hx'] * Grid['hy'] * Grid['hz']
        Grid['K'] = np.ones((Grid['Nz'], Grid['Ny'], Grid['Nx'], 3))
        Grid['K_alpha'] = np.ones((Grid['Nz'], Grid['Ny'], Grid['Nx'],
    3))
        Grid['por'] = np.ones((Grid['Nz'], Grid['Ny'], Grid['Nx']))
        # porosity tensor of the problem
        Grid['por_alpha'] = .28 * np.ones((Grid['Nz'], Grid['Ny'], Grid
    ['Nx']))
        Fluid = {}   # fluid properties
        Fluid['vw'] = 1.
        Fluid['vo'] = 5.
        Fluid['swc'] = 0.2
        Fluid['sor'] = 0.2
        Q = np.zeros(N)  # source and sink
        Q[[0, N - 1]] = [.1, -.1] # source and sink values
        s_initial = np.ones(N) * 0.2  # initial saturation
        dt = 0.05  # time step size
        nt = 20  # number of time steps
        NewtRaph_IT = 3
        self.Grid = Grid
        self.N, self.dt, self.nt = N, dt, nt
        self.Fluid = Fluid
        self.Q, self.s_initial, self.NewtRaph_IT = Q, s_initial,
    NewtRaph_IT
        self.Nbasis = 50
```

Listing A.1: Setting input parameters

```
# in the name of almighty
# Assalamu Alaikum
from __future__ import  division
import numpy as np
import matplotlib.pyplot as plt
import scipy
from scipy.sparse import spdiags
```

```
from scipy.sparse.linalg import spsolve
from scipy.sparse.linalg import cg
import scipy.io
from ONEparam import Params


class Pressure(object):
    def __init__(self):
        param = Params()
        self.Grid = param.Grid
        self.N = param.N
        self.Fluid = param.Fluid
        self.Q = param.Q

    def relPerm(self, s):
        '''
        this function is similar to relPerm in saturation class
        input--> s saturation field at present time step
        output--> mobility of water,oil
        '''
        Fluid = self.Fluid
        S = (s - Fluid['swc']) / (1.0 - Fluid['swc'] - Fluid['sor'])
        Mw = np.square(S) / Fluid['vw']
        Mo = np.square(1.0 - S) / Fluid['vo']
        dMw = 2.0 * S / Fluid['vw'] / \
              (1.0 - Fluid['swc'] - Fluid['sor'])
        dMo = -2.0 * (1.0 - S) / Fluid['vo'] / \
              (1.0 - Fluid['swc'] - Fluid['sor'])
        return Mw, Mo, dMw, dMo

    def tpfa(self, K, dirichlet=None):
        '''
        solve the pressure equation
        input--> K is permeability, function of saturation field
        output--> V velocity stored in the dictionary
        '''
        Grid = self.Grid
```

```
45          q0 = self.Q
46          # Compute transmissibilities by harmonic averaging.
47          Nx = Grid['Nx']
48          hx = Grid['hx']
49          Ny = Grid['Ny']
50          hy = Grid['hy']
51          Nz = Grid['Nz']
52          hz = Grid['hz']
53          N = self.N # Nx * Ny * Nz
54          L = 1.0 / K
55          tx = 2 * hy * hz / hx
56          TX = np.zeros((Nz, Ny, Nx + 1))
57          ty = 2 * hx * hz / hy
58          TY = np.zeros((Nz, Ny + 1, Nx))
59          tz = 2 * hx * hy / hz
60          TZ = np.zeros((Nz + 1, Ny, Nx))
61          TX[:, :, 1:Nx] = tx / (L[:, :, 0:Nx - 1, 0] + L[:, :, 1:Nx, 0])
62          TY[:, 1:Ny, :] = ty / (L[:, 0:Ny - 1, :, 1] + L[:, 1:Ny, :, 1])
63          TZ[1:Nz, :, :] = tz / (L[0:Nz - 1, :, :, 2] + L[1:Nz, :, :, 2])
64
65          # Assemble TPFA discretization matrix.
66          x1 = np.reshape(TX[:, :, 0:Nx], N)
67          x2 = np.reshape(TX[:, :, 1:Nx + 1], N)
68          y1 = np.reshape(TY[:, 0:Ny, :], N)
69          y2 = np.reshape(TY[:, 1:Ny + 1, :], N)
70          z1 = np.reshape(TZ[0:Nz, :, :], N)
71          z2 = np.reshape(TZ[1:Nz + 1, :, :], N)
72          DiagVecs = np.array([-z2, -y2, -x2, x1 + x2 + y1 + y2 +
73                               z1 + z2, -x1, -y1, -z1])
74          DiagIndx = np.array([-Nx * Ny, -Nx, -1, 0, 1, Nx, Nx * Ny])
75
76          # Eliminate any zero vectors present
77          nonzero_diags_indices = ~np.all(DiagVecs == 0, axis=1)
78          DiagVecs = DiagVecs[nonzero_diags_indices]
79          DiagIndx = DiagIndx[nonzero_diags_indices]
80          # assemble the sparse matrix
81          A = spdiags(DiagVecs, DiagIndx, N, N).tocsr()
```

```
82          # print 'SALAM'
83          # self.genA_full(x1, x2, y1, y2, z1, z2)
84          q = np.copy(q0)
85          # Impose boundary conditions.
86          if dirichlet is not None:
87              large_number = 1e12
88              idxs = dirichlet[:, 0].astype('int')
89              vals = dirichlet[:, 1]
90              A[idxs, idxs] = large_number
91              q[idxs] = large_number * vals
92          else:
93              A[0, 0] = A[0, 0] + sum(Grid['K_alpha'][:, 0, 0, 0])
94
95          # Solve linear system and extract interface fluxes.
96          u = spsolve(A, q)
97          # print 'salam u shape \n  {} '.format(A.todense()[::50, ::50])
98          # u_c = self.conjugate_grad(A=A, b=q)
99          print 'salam error in cg  \n {}'.format(np.shape(q))
100         P = np.reshape(u, (Nz, Ny, Nx))
101         #
102         V = dict()
103         V['x'] = np.zeros((Nz, Ny, Nx + 1))
104         V['y'] = np.zeros((Nz, Ny + 1, Nx))
105         V['z'] = np.zeros((Nz + 1, Ny, Nx))
106         V['x'][:, :, 1:Nx] = (P[:, :, 0:Nx - 1] -
107                               P[:, :, 1:Nx]) * TX[:, :, 1:Nx]
108         V['y'][:, 1:Ny, :] = (P[:, 0:Ny - 1, :] -
109                               P[:, 1:Ny, :]) * TY[:, 1:Ny, :]
110         V['z'][1:Nz, :, :] = (P[0:Nz - 1, :, :] -
111                               P[1:Nz, :, :]) * TZ[1:Nz, :, :]
112         return u, V
113
114     def pres(self, S):
115         '''
116         - compute mobility functions by calling relPerm function
117           which depends on s
118         - call tpfa to solve the pressure equation
```

```
        input--> S  saturation field at present time step
        output-->pressure and velocity stored in the dictionary
        '''
        Grid = self.Grid
        Mw, Mo, dMw, dMo = self.relPerm(S)
        Mt = Mw + Mo
        KM = np.reshape(np.array([Mt, Mt, Mt]).T,
                        (Grid['Nz'], Grid['Ny'],
                         Grid['Nx'], 3)) * Grid['K_alpha']
        return self.tpfa(KM)
```

Listing A.2: Pressure class

```
# in the name of almighty
# Assalamu Alaikum
from __future__ import division
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import spdiags
from scipy.sparse.linalg import spsolve
import scipy
import scipy.io
from scipy.sparse.linalg import bicgstab, gmres, bicg, minres
from scipy.sparse import csr_matrix
import cPickle as pickle
from ONEparam import Params


class Saturation(object):

    def __init__(self):
        param = Params()
        self.Grid = param.Grid
        self.N = param.N
        self.dt = param.dt
        self.nt = param.nt
        self.Fluid = param.Fluid
        self.Q = param.Q
```

```
26          self.NewtRaph_IT = param.NewtRaph_IT
27          self.IT_numsteps = np.ceil(np.power(2, self.NewtRaph_IT))
28          self.Nbasis = param.Nbasis
29
30      def relPerm(self, s):
31          '''
32          this function is similar to relPerm
33          input--> s saturation field at present time step
34          output--> mobility of water,oil
35          '''
36          Fluid = self.Fluid
37          S = (s - Fluid['swc']) / (1.0 - Fluid['swc'] - Fluid['sor'])
38          Mw = np.square(S) / Fluid['vw']
39          Mo = np.square(1.0 - S) / Fluid['vo']
40          dMw = 2.0 * S / Fluid['vw'] / \
41                (1.0 - Fluid['swc'] - Fluid['sor'])
42          dMo = -2.0 * (1.0 - S) / Fluid['vo'] / \
43                (1.0 - Fluid['swc'] - Fluid['sor'])
44          return Mw, Mo, dMw, dMo
45
46      def genA(self, V):  # Return A
47          '''
48          assemble sparse matrix to solve saturation using upwind scheme
49          input--> V velocity vector stored in dictionary
50          output--> sparse matrix A to solve saturation equation
51          '''
52          Grid = self.Grid
53          q = self.Q
54          Nx = Grid['Nx']
55          Ny = Grid['Ny']
56          Nz = Grid['Nz']
57          N = Nx * Ny * Nz
58          fp = np.minimum(q, 0)
59          XN = np.minimum(V['x'], 0)
60          x1 = np.reshape(XN[:, :, 0:Nx], N)
61          YN = np.minimum(V['y'], 0)
62          y1 = np.reshape(YN[:, 0:Ny, :], N)
```

```
63          ZN = np.minimum(V['z'], 0)
64          z1 = np.reshape(ZN[0:Nz, :, :], N)
65          XP = np.maximum(V['x'], 0)
66          x2 = np.reshape(XP[:, :, 1:Nx + 1], N)
67          YP = np.maximum(V['y'], 0)
68          y2 = np.reshape(YP[:, 1:Ny + 1, :], N)
69          ZP = np.maximum(V['z'], 0)
70          z2 = np.reshape(ZP[1:Nz + 1, :, :], N)
71          DiagVecs = np.array([z2, y2, x2, fp + x1 - x2 +
72                               y1 - y2 + z1 - z2, -x1, -y1, -z1])
73          DiagIndx = np.array([-Nx * Ny, -Nx, -1, 0, 1, Nx, Nx * Ny])
74          # Eliminate any zero vectors present
75          nonzero_diags_indices = ~np.all(DiagVecs == 0, axis=1)
76          DiagVecs = DiagVecs[nonzero_diags_indices]
77          DiagIndx = DiagIndx[nonzero_diags_indices]
78          A = spdiags(DiagVecs, DiagIndx, N, N).tocsr()
79
80          return A
81
82
83      def upstream(self, S, V):  # Return S
84          '''
85          advect the saturation using explicit time discretization scheme
    .
86          time_step determined by cfl condition for numerical stability
87          velocity is kept constant during this inner time steps
88          input--> S saturation value at the present time step
89          input--> V velocity stored in dictionary
90          output--> smatrix saturation matrix for Nts time steps
91          '''
92          Grid = self.Grid
93          Fluid = self.Fluid
94          q = self.Q
95          dt = self.dt
96          Nx = Grid['Nx']
97          Ny = Grid['Ny']
98          Nz = Grid['Nz']
```

```
99          N = self.N # Nx * Ny * Nz
100         # Volume*Porosity
101         pv = Grid['V'] * np.reshape(Grid['por_alpha'], N)
102         # injection rate
103         fi = np.maximum(q, 0)
104
105         # compute cfl condition
106         XP = np.maximum(V['x'], 0)
107         XN = np.minimum(V['x'], 0)
108         YP = np.maximum(V['y'], 0)
109         YN = np.minimum(V['y'], 0)
110         ZP = np.maximum(V['z'], 0)
111         ZN = np.minimum(V['z'], 0)
112         # Total flux into each block
113         Vi = XP[:, :, 0:Nx] + YP[:, 0:Ny, :] + ZP[0:Nz, :, :] \
114               - XN[:, :, 1:Nx + 1] - YN[:, 1:Ny + 1, :] \
115               - ZN[1:Nz + 1, :, :]
116         Vi = np.reshape(Vi, N)
117         # print 'Vi',Vi
118         pm = np.min(pv / (Vi + fi))
119         cfl = ((1.0 - Fluid['swc'] - Fluid['sor']) / 3)\
120                * pm
121         Nts = np.ceil(dt / cfl)
122         dtx = (dt / Nts) / pv
123
124         # assemble advection matrix A
125         A = self.genA(V)
126         # Compute A*dt/|Omega_i|
127         A = spdiags(dtx, 0, N, N).dot(A)
128         # Compute Q_in*dt/|Omega_i|
129         fi = fi * dtx
130         print 'salam full Nts = ', Nts
131         # inner time iterations for Nts time steps
132         smatrix = np.ones((Nts, N))
133         for t in np.arange(0, Nts):
134             mw, mo, dmw, dmo = self.relPerm(S)
135             fw = mw / (mw + mo)
```

```
136             S = S + (A.dot(fw) + fi)
137             smatrix[t] = S
138 
139         return smatrix
140 
141     def newtRaph(self, S, V):  # Return S
142         '''
143         Newton Raphson method to solve system of nonlinear equations
144         input--> S saturation at present time step
145         input--> V velocity stored in dictionary
146         output--> saturation saturation matrix for IT_numsteps time
    steps
147         '''
148         Grid = self.Grid
149         Fluid = self.Fluid
150         q = self.Q
151         T = self.dt
152         NewtRaph_IT = self.NewtRaph_IT
153         IT_numsteps = self.IT_numsteps
154 
155         N = Grid['Nx'] * Grid['Ny'] * Grid['Nz']
156         ###################################################################
157         # generating A matrix
158         A = self.genA(V)
159         ###################################################################
160         conv = 0
161         IT = NewtRaph_IT
162         S00 = S
163         # saturation = np.ones((IT_numsteps, N))
164         saturation = []
165         while conv == 0:
166             dt = T / np.power(2, IT)
167             # Volume*Porosity
168             pv = Grid['V'] * np.reshape(Grid['por_alpha'], N)
169             dtx = dt / pv
170             ###############################################
171             # Compute A*dt/|Omega_i|
```

```
172             B = spdiags(dtx, 0, N, N).dot(A)
173             ##################################################
174             # Compute Q_in*dt/|Omega_i|
175             fi = np.maximum(q, 0) * dtx
176             ##################################################
177             # saturation = np.ones((IT_numsteps, N))
178             saturation = []
179             fwlist = []
180             I = 0
181             tol = 1e-3
182             k = 0
183             dsn = 1
184             while I < np.power(2, IT):
185                 S0 = S
186                 dsn = 1
187                 it = 0
188                 I = I + 1
189
190                 while dsn > tol and it < 10:
191                     Mw, Mo, dMw, dMo = self.relPerm(S)
192                     df = dMw / (Mw + Mo) - Mw / \
193                         np.power((Mw + Mo), 2) * (dMw + dMo)
194                     dG = scipy.sparse.identity(N) - \
195                         B.dot(spdiags(df, 0, N, N))
196                     fw = Mw / (Mw + Mo)
197                     G = S - S0 - (B.dot(fw) + fi)
198                     # print 'salam B dot fw shape',
199                     # B.dot(fw).shape, np.dot(B.todense(), fw)
200                     dS = spsolve(-dG, G)
201                     # dS, _ = bicgstab(-dG, G)
202                     # dS, _ = gmres(-dG, G, M=-dG)
203                     # dS, _ = bicg(-dG, G)
204                     # dS, _ = minres(-dG, G, M=-dG)
205
206                     S = S + dS
207
208                     dsn = np.linalg.norm(dS)
```

```
                    it = it + 1
                # print 'salam it {} dsn {} '.format(it, dsn)
                if dsn > tol:
                    I = np.power(2, IT)
                    S = S00
                # saturation[k, :] = S
                saturation.append(S)
                fwlist.append(fw)
                k = k + 1

            if dsn < tol:
                conv = 1
            else:
                IT = IT + 1
                # conv=1
        print 'salam NewtRaph time steps {}'.format(IT)
        return np.asarray(saturation)
```

Listing A.3: Saturation class

```
# in the name of almighty
# Assalamu Alaikum
import numpy as np
import matplotlib.pyplot as plt
import scipy
import  scipy.io
import time
import cPickle as pickle
import prettyplotlib as pplt
import multiprocessing as mp
from ONEparam import Params, genpermeability
from ONEpressure import Pressure
from ONEsaturation import Saturation


def runq5(K):
    '''
    - solve for pressure and saturation
```

```
19    - call functions in pressure class and functions in saruration
      class
20    - solve using implcit NewtRaph method
21    Input ---> K is the permeability tensor or order 4 (1, Nx, Ny, 3)
22
23    Output ---> listsnapshot = [u1_matrix, sat_matrix, u2list, fwlist,
      Mtmatrix]
24    u1_matrix  is a pressure solution matrix of size (nt x N)
25    sat_matrix is a matrix of size of (nt x N)
26    containing the last solution saturation vector of each element in
      the u2list
27    fwlist is same mentioned earlier
28    u2list  is a list of size nt , each element in the list
29    containing a saturation solution matrix of size (Nts x N).
30    '''
31    param = Params()
32    pressure = Pressure()
33    saturation = Saturation()
34    nt, N, s = param.nt, param.N, param.s_initial
35
36    sat_matrix = np.ones((nt, N), dtype=np.float32)
37    u1_matrix = np.ones((nt, N), dtype=np.float32)
38    u2list = []
39    for t in xrange(nt):
40        # solve for pressure
41        u, V = pressure.pres(s, K)
42        # solve for saturation
43        # smatrix = saturation.upstream(s, V)
44        smatrix = saturation.newtRaph(s, V)
45        s = smatrix[np.int(Nts)-1]
46        # pack the data
47        sat_matrix[t] = np.asarray(s, dtype=np.float32)
48        u1_matrix[t] = np.asarray(u, dtype=np.float32)
49        u2list.append(np.asarray(smatrix, dtype=np.float32))
50
51    listsnapshot = [u1_matrix, sat_matrix, u2list]
52
```

```
53      return listsnapshot
54
55
56  def gendatasetLarsElden(rank, i, filetype='svdtrain'):
57      print 'salam preparing permeability set ...'
58      filename = 'ONEperm_real_sqrt_64_64_' + str(rank+1) + '.mat'
59      perm_real_dict = scipy.io.loadmat(filename)
60      perm_real = perm_real_dict['perm_realidx']
61      print 'salam rank {} permeability shape {}'.format(rank+1,
        perm_real.shape)
62      K = np.ones((1, 64, 64, 3))
63      permgrid = np.reshape(perm_real[i, :], (64, 64), order='F')
64      K[0, :, :, 0] = permgrid
65      K[0, :, :, 1] = permgrid
66      K[0, :, :, 2] = permgrid
67      ###############################################################
68      Listsnapshot = runq5(K)
69      ###############################################################
70      set = rank + 1
71      filename = 'ONElistsnapshot' + filetype + str(set) + '.pkl'
72      fileobject = open(filename, 'wb')
73      pickle.dump(Listsnapshot, fileobject)
74      fileobject.close()
75
76
77  if __name__ == '__main__':
78      t0 = time.time()
79      #
        ###########################################################################

80      print               'salam parallel runs  ...'
81      #
        ###########################################################################

82      svdtrain, nonsvdtrain, test = [0, 'svdtrain'], [1, 'nonsvdtrain'],
        [44, 'test']
83      processes = []
```

```
84      num_processes = 45
85      for rank in range(num_processes):
86          p = mp.Process(target=gendatasetLarsElden, args=(rank, test[0],
        test[1]))
87          p.start()
88          processes.append(p)
89      for p in processes:
90          p.join()
91      print 'salam Elapsed time %f '% (time.time() - t0)
```

Listing A.4: Main function

Bibliography

[1] Jørg E Aarnes, Tore Gimse, and Knut A Lie. An introduction to the numerics of flow in porous media using Matlab. In *Geometric modelling, numerical simulation, and optimization*, pages 265–306. Springer, 2007.

[2] Mahdi Abdi-Khanghah, Amin Bemani, Zahra Naserzadeh, and Zhien Zhang. Prediction of solubility of n-alkanes in supercritical co 2 using rbf-ann and mlp-ann. *Journal of CO2 Utilization*, 25:108–119, 2018.

[3] Assyr Abdulle, Andrea Barth, and Christoph Schwab. Multilevel monte carlo methods for stochastic elliptic multiscale pdes. *Multiscale Modeling & Simulation*, 11(4):1033–1070, 2013.

[4] Zeid M Alghareeb and John Williams. Optimum decision-making in reservoir managment using reduced-order models. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2013.

[5] Athanasios C Antoulas, Danny C Sorensen, and Serkan Gugercin. A survey of model reduction methods for large-scale systems. *Contemporary mathematics*, 280:193–220, 2001.

[6] Michael J Asher, Barry FW Croke, Anthony J Jakeman, and Luk JM Peeters. A review of surrogate models and their application to groundwater modeling. *Water Resources Research*, 51(8):5957–5973, 2015.

[7] Patricia Astrid. *Reduction of process simulation models: a proper orthogonal decomposition approach*. Technische Universiteit Eindhoven Ph. D.-thesis, 2004.

[8] David A.Wood. transparent open-box learning network provides insight to complex systems and a performance benchmark for more-opaque machine learning algorithms. *Advances in Geo-Energy Research*, 2(2):148–162, 2018.

[9] Masoud Babaei, Ahmed H. Elsheikh, and Peter R. King. A comparison study between an adaptive quadtree grid and uniform grid upscaling for reservoir simulation. *Transport in Porous Media*, 98:377–400, 2013. URL `http://dx.doi.org/10.1007/s11242-013-0149-7`.

[10] Ivo Babuška, Fabio Nobile, and Raul Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45(3):1005–1034, 2007.

[11] Coryn AL Bailer-Jones, David JC MacKay, and Philip J Withers. A recurrent neural network for modelling dynamical systems. *network: computation in neural systems*, 9(4):531–547, 1998.

[12] Maxime Barrault, Yvon Maday, Ngoc C Nguyen, and Anthony T. Patera. An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667 – 672, 2004. ISSN 1631-073X. URL `http://dx.doi.org/10.1016/j.crma.2004.08.006`.

[13] Andrea Barth, Christoph Schwab, and Nathaniel Zollinger. Multi-level monte carlo finite element method for elliptic pdes with stochastic coefficients. *Numerische Mathematik*, 119(1):123–161, 2011.

[14] Peter Bastian. *Numerical computation of multiphase flow in porous media*. PhD thesis, habilitationsschrift Univeristät Kiel, 1999.

[15] Domenico A Baú. Planning of groundwater supply systems subject to uncertainty using stochastic flow reduced models and multi-objective evolutionary optimization. *Water resources management*, 26(9):2513–2536, 2012.

[16] Hamid Bazargan, Mike Christie, Ahmed H. Elsheikh, and Mohammad Ahmadi. Surrogate accelerated sampling of reservoir models with complex struc-

tures using sparse polynomial chaos expansion. *Advances in Water Resources*, 86:385–399, 2015. URL `http://dx.doi.org/10.1016/j.advwatres.2015.09.009`.

[17] Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

[18] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[19] Ilias Bilionis and Nicholas Zabaras. Multi-output local gaussian process regression: Applications to uncertainty quantification. *Journal of Computational Physics*, 231(17):5718–5746, 2012.

[20] Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

[21] Scott E Boyce and William WG Yeh. Parameter-independent model reduction of transient groundwater flow models: Application to inverse problems. *Advances in water resources*, 69:168–180, 2014.

[22] Marcelo Buffoni and Karen Willcox. Projection-based model reduction for reacting flows. In *40th Fluid Dynamics Conference and Exhibit*, page 5008, 2010.

[23] Tan Bui-Thanh, Murali Damodaran, and Karen E Willcox. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA journal*, 42(8):1505–1516, 2004.

[24] Tan Bui-Thanh, Karen Willcox, Omar Ghattas, and Bart VB Waanders. Goal-oriented, model-constrained optimization for reduction of large-scale systems. *Journal of Computational Physics*, 224(2):880–896, 2007.

[25] John A Bullinaria. Recurrent neural networks. *Neural Computation: Lecture*, 12, 2013.

[26] Yanhua Cao, Jiang Zhu, Zhendong Luo, and Ionel M Navon. Reduced-order modeling of the upper tropical pacific ocean model using proper orthogonal

decomposition. *Computers & Mathematics with Applications*, 52(8-9):1373–1386, 2006.

[27] Marco A Cardoso and Louis J Durlofsky. Linearized reduced-order models for subsurface flow simulation. *Journal of Computational Physics*, 229(3): 681–700, 2010.

[28] Marco A Cardoso, Louis J Durlofsky, and Pallav Sarma. Development and application of reduced-order modeling procedures for subsurface flow simulation. *International journal for numerical methods in engineering*, 77(9):1322–1350, 2009.

[29] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

[30] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32 (5):2737–2764, 2010.

[31] Saifon Chaturantabut and Danny C Sorensen. Application of POD and DEIM on dimension reduction of non-linear miscible viscous fingering in porous media. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4): 337–353, 2011.

[32] Saifon Chaturantabut and Danny C Sorensen. A state space error estimate for pod-deim nonlinear model reduction. *SIAM Journal on numerical analysis*, 50(1):46–63, 2012.

[33] Y Chen, Louis J Durlofsky, M Gerritsen, and Xian-Huan Wen. A coupled local–global upscaling approach for simulating flow in highly heterogeneous formations. *Advances in Water Resources*, 26(10):1041–1060, 2003.

[34] Zhangxin Chen, Guanren Huan, and Yuanle Ma. *Computational methods for multiphase flows in porous media*. SIAM, 2006.

[35] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[36] K Andrew Cliffe, Mike B Giles, Robert Scheichl, and Aretha L Teckentrup. Multilevel monte carlo methods and applications to elliptic pdes with random coefficients. *Computing and Visualization in Science*, 14(1):3, 2011.

[37] Ferran V Codina, Nguyen C Ngoc, Michael B Giles, and Jaime Peraire. A model and variance reduction method for computing statistical outputs of stochastic elliptic partial differential equations. *Journal of Computational Physics*, 297:700–720, 2015.

[38] Ferran V Codina, Nguyen C Ngoc, Mike B Giles, and Jaime Peraire. An empirical interpolation and model-variance reduction method for computing statistical outputs of parametrized stochastic partial differential equations. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):244–265, 2016.

[39] Wim De Mulder, Steven Bethard, and Marie-Francine Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61–98, 2015.

[40] RJ Dedden. Model order reduction using the discrete empirical interpolation method. 2012.

[41] Alireza Doostan and Houman Owhadi. A non-adapted sparse approximation of pdes with stochastic inputs. *Journal of Computational Physics*, 230(8): 3015–3034, 2011.

[42] Louis J Durlofsky. Upscaling and gridding of fine scale geological models for flow simulation. In *8th International Forum on Reservoir Simulation Iles Borromees, Stresa, Italy*, volume 2024, 2005.

[43] Louis J Durlofsky and Yuguang Chen. Uncertainty quantification for subsurface flow problems using coarse-scale models. In *Numerical analysis of multiscale problems*, pages 163–202. Springer, 2012.

[44] Y Efendiev, Oleg Iliev, and C Kronsbein. Multilevel monte carlo methods using ensemble level mixed msfem for two-phase flow and transport simulations. *Computational geosciences*, 17(5):833–850, 2013.

[45] Yalchin Efendiev, Bangti Jin, Presho Michael, and Xiaosi Tan. Multilevel markov chain monte carlo method for high-contrast single-phase flow problems. *Communications in Computational Physics*, 17(1):259–286, 2015.

[46] Lars Eldén. *Matrix methods in data mining and pattern recognition*, volume 4. SIAM, 2007.

[47] Doaa Elsakout, Mike Christie, Gabriel Lord, et al. Multilevel markov chain monte carlo (mlmcmc) for uncertainty quantification. In *SPE North Africa Technical Conference and Exhibition*. Society of Petroleum Engineers, 2015.

[48] Ahmed H. Elsheikh, Matthew Jackson, and Tara Laforce. Bayesian reservoir history matching considering model and parameter uncertainties. *Mathematical Geosciences*, 44(5):515–543, Jul 2012. ISSN 1874-8953. URL `https://doi.org/10.1007/s11004-012-9397-2`.

[49] Ahmed H. Elsheikh, Mary F. Wheeler, and Ibrahim Hoteit. Nested sampling algorithm for subsurface flow model selection, uncertainty quantification, and nonlinear calibration. *Water Resources Research*, 49(12):8383–8399, 2013. ISSN 1944-7973. URL `http://dx.doi.org/10.1002/2012WR013406`.

[50] Ahmed H. Elsheikh, Ibrahim Hoteit, and Mary F. Wheeler. Efficient bayesian inference of subsurface flow models using nested sampling and sparse polynomial chaos surrogates. *Computer Methods in Applied Mechanics and Engineering*, 269:515–537, 2014. URL `http://dx.doi.org/10.1016/j.cma.2013.11.001`.

[51] Fritjof Fagerlund, Fredrik Hellman, Axel Målqvist, and Auli Niemi. Multilevel monte carlo methods for computing failure probability of porous media flow systems. *Advances in water resources*, 94:498–509, 2016.

[52] F. Fang, C.C. Pain, I.M. Navon, A.H. Elsheikh, J. Du, and D. Xiao. Non-linear petrovgalerkin methods for reduced order hyperbolic equations and discontinuous finite element methods. *Journal of Computational Physics*, 234:540–559, 2013. URL http://dx.doi.org/10.1016/j.jcp.2012.10.011.

[53] Michalis Frangos, Youssef Marzouk, Karen Willcox, and Bart VB Waanders. *Surrogate and Reduced-Order Modeling: A Comparison of Approaches for Large-Scale Statistical Inverse Problems*, pages 123–149. John Wiley & Sons, Ltd, 2010. ISBN 9780470685853. URL http://dx.doi.org/10.1002/9780470685853.ch7.

[54] Roland W Freund. Model reduction methods based on Krylov subspaces. *Acta Numerica*, 12:267–319, 2003.

[55] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[56] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6): 801–806, 1993.

[57] Lynn W Gelhar. Stochastic subsurface hydrology from theory to applications. *Water Resources Research*, 22(9S):135S–145S, 1986.

[58] G Geraci, G Iaccarino, F Montomoli, et al. Multi-fidelity uncertainty quanti cation using rans and dns.

[59] G Geraci, M Eldred, and G Iaccarino. A multifidelity control variate approach for the multilevel monte carlo technique. 2015.

[60] Roger G Ghanem and Pol D Spanos. Stochastic finite element method: Response statistics. In *Stochastic finite elements: a spectral approach*, pages 101–119. Springer, 1991.

[61] Mohammadreza Ghasemi. *Model order reduction in porous media flow simulation and optimization*. Ph.D. thesis, Texas AM Univeristy, 2015.

[62] Michael B Giles. Multilevel monte carlo path simulation. *Operations Research*, 56(3):607–617, 2008.

[63] Michael B Giles. Multilevel monte carlo methods. In *Monte Carlo and Quasi-Monte Carlo Methods 2012*, pages 83–103. Springer, 2013.

[64] Michael B Giles. Multilevel monte carlo methods. *Acta Numerica*, 24:259–328, 2015.

[65] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[66] Serkan Gugercin and Athanasios C Antoulas. A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77(8):748–766, 2004.

[67] Jincong He. *Enhanced linearized reduced-order models for subsurface flow simulation*. M.S. thesis, Stanford Univeristy, 2010.

[68] Jincong He. *Reduced-order modeling for oil-water and compositional systems, with application to data assimilation and production optimization*. Ph.D. thesis, Stanford Univeristy, 2013.

[69] Jincong He, Jonsat Sætrom, and Louis J Durlofsky. Enhanced linearized reduced-order models for subsurface flow simulation. *Journal of Computational Physics*, 230(23):8313–8341, 2011.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[71] T Heijn, Renato Markovinovic, and Jan D Jansen. Generation of low-order reservoir models using system-theoretical concepts. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2003.

[72] Stefan Heinrich. Multilevel monte carlo methods. In Svetozar Margenov, Jerzy Waśniewski, and Plamen Yalamov, editors, *Large-Scale Scientific Computing*,

pages 58–67, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-45346-8.

[73] Michiel Hermans and Benjamin Schrauwen. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198, 2013.

[74] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, and Tara N Sainath. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[75] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[76] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[77] LY Hu and T Chugunova. Multiple-point geostatistics for modeling subsurface heterogeneity: A comprehensive review. *Water Resources Research*, 44(11), 2008.

[78] Fayadhoi Ibrahima. *Probability distribution methods for nonlinear transport in heterogenous porous media.* Ph.D. thesis, Stanford Univeristy, 2016.

[79] Ozan Irsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728, 2014.

[80] J-D Jansen, DR Brouwer, G Naevdal, and CPJW Van Kruijsdijk. Closed-loop reservoir management. *First Break*, 23(1):43–48, 2005.

[81] Jan Dirk Jansen and Louis J Durlofsky. Use of reduced-order models in well control optimization. *Optimization and Engineering*, 18(1):105–132, 2017.

[82] Zhaoyang L Jin and Louis J Durlofsky. Reduced-order modeling of co 2 storage operations. *International Journal of Greenhouse Gas Control*, 68:49–67, 2018.

[83] Laureline Josset, Vasily Demyanov, Ahmed H. Elsheikh, and Ivan Lunati. Accelerating Monte Carlo Markov chains with proxy and error models. *Computers & Geosciences*, 85:38–48, 2015. URL `https://doi.org/10.1016/j.cageo.2015.07.003`.

[84] J Nagoor Kani and Ahmed H Elsheikh. Reduced-order modeling of subsurface multi-phase flow models using deep residual recurrent neural networks. *Transport in Porous Media*, pages 1–29, 2018.

[85] Ahmed Kebaier et al. Statistical romberg extrapolation: a new variance reduction method and applications to option pricing. *The Annals of Applied Probability*, 15(4):2681–2705, 2005.

[86] Robert C Knox. *Subsurface Transport and Fate Processes: 0*. CRC Press, 2018.

[87] Slawomir Koziel and Leifur Leifsson. Surrogate-based modeling and optimization. *Applications in Engineering*, 2013.

[88] Sanjay Lall, Jerrold E Marsden, and Sonja Glavaški. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International journal of robust and nonlinear control*, 12(6):519–535, 2002.

[89] Toni Lassila, Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Model order reduction in fluid dynamics: challenges and perspectives. In *Reduced Order Methods for modeling and computational reduction*, pages 235–273. Springer, 2014.

[90] Hao Li, Zhien Zhang, and Zhijian Liu. Application of artificial neural networks for catalysis: a review. *Catalysts*, 7(10):306, 2017.

[91] Heng Li and Dongxiao Zhang. Probabilistic collocation method for flow in

porous media: Comparisons with other stochastic methods. *Water Resources Research*, 43(9), 2007.

[92] Binghuai Lin. *Reduced-order modeling for ensemble real-time estimation and control*. PhD thesis, Massachusetts Institute of Technology, 2012.

[93] Guang Lin and Alexandre M Tartakovsky. An efficient, high-order probabilistic collocation method on sparse grids for three-dimensional flow and solute transport in randomly heterogeneous porous media. *Advances in Water Resources*, 32(5):712–722, 2009.

[94] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[95] Dan Lu, Guannan Zhang, Clayton Webster, and Charlotte Barbier. An improved multilevel monte carlo method for estimating probability distribution functions in stochastic oil reservoir simulations. *Water Resources Research*, 52(12):9642–9660, 2016.

[96] David J Lucia, Philip S Beran, and Walter A Silva. Reduced-order modeling: new approaches for computational physics. *Progress in Aerospace Sciences*, 40 (1):51–117, 2004.

[97] John L Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, 1967.

[98] James McPhee and William WG Yeh. Groundwater management using model reduction via empirical orthogonal functions. *Journal of Water Resources Planning and Management*, 134(2):161–170, 2008.

[99] Marcus Meyer and Hermann G Matthies. Efficient model reduction in non-linear dynamics using the karhunen-loéve expansion and dual-weighted-residual methods. *Computational Mechanics*, 31(1-2):179–191, 2003.

[100] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

[101] Siddhartha Mishra, Ch Schwab, and Jonas Šukys. Multi-level monte carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions. *Journal of Computational Physics*, 231(8):3365–3388, 2012.

[102] Siddhartha Mishra, Ch Schwab, and Jonas Šukys. Multi-level monte carlo finite volume methods for uncertainty quantification of acoustic wave propagation in random heterogeneous layered medium. *Journal of Computational Physics*, 312:192–217, 2016.

[103] Florian Müller, Patrick Jenny, and Daniel W Meyer. Multilevel monte carlo for two phase flow and buckley–leverett transport in random heterogeneous porous media. *Journal of Computational Physics*, 250:685–702, 2013.

[104] J. Nagoor Kani and A. H. Elsheikh. DR-RNN: A deep residual recurrent neural network for model reduction. *ArXiv e-prints*, September 2017.

[105] Leo Wai-Tsun Ng. *Multifidelity approaches for design under uncertainty*. PhD thesis, Massachusetts Institute of Technology, 2013.

[106] Ngoc C Nguyen, Anthony T. Patera, and Jaime Peraire. A best points interpolation method for efficient approximation of parametrized functions. *International journal for numerical methods in engineering*, 73(4):521–543, 2008.

[107] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[108] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning (3)*, 28:1310–1318, 2013.

[109] Damiano Pasetto, Alberto Guadagnini, and Mario Putti. POD-based Monte-Carlo approach for the solution of regional scale groundwater flow driven by randomly distributed recharge. *Advances in water resources*, 34(11):1450–1463, 2011.

[110] Damiano Pasetto, Mario Putti, and William WG Yeh. A reduced-order model for groundwater flow equation with random hydraulic conductivity: Application to Monte-Carlo methods. *Water Resources Research*, 49(6):3215–3228, 2013.

[111] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[112] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[113] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Optimal model management for multifidelity monte carlo estimation. *SIAM Journal on Scientific Computing*, 38(5):A3163–A3194, 2016.

[114] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018.

[115] Kurt R. Petvipusit, Ahmed H. Elsheikh, Tara C. Laforce, Peter R. King, and Martin J. Blunt. Robust optimisation of CO2 sequestration strategies under geological uncertainty using adaptive sparse grid surrogates. *Computational Geosciences*, 18(5):763–778, Oct 2014. ISSN 1573-1499. URL `https://doi.org/10.1007/s10596-014-9425-z`.

[116] Richard H Pletcher, John C Tannehill, and Dale Anderson. *Computational fluid mechanics and heat transfer*. CRC Press, 2012.

[117] Justyna Katarzyna Przybysz-Jarnut. Hydrocarbon reservoir parameter estimation using production data and time-lapse seismic. 2010.

[118] Saman Razavi, Bryan A Tolson, and Donald H Burn. Review of surrogate modeling in water resources. *Water Resources Research*, 48(7), 2012.

[119] Michal Rewienski and Jacob White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 22(2):155–170, 2003.

[120] Matthieu AH Rousset, Chung K Huang, Hector Klie, and Louis J Durlofsky. Reduced-order modeling for thermal recovery processes. *Computational Geosciences*, 18(3-4):401–415, 2014.

[121] Gianluigi Rozza, Dinh BP Huynh, and Anthony T Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3), 2007. URL `https://doi.org/10.1007/BF03024948`.

[122] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[123] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[124] Adam J Siade, Mario Putti, and William WG Yeh. Snapshot selection for groundwater model reduction using proper orthogonal decomposition. *Water Resources Research*, 46(8):W08539, 2010.

[125] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of applied mathematics*, 45(3):561–571, 1987.

[126] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pages 2171–2180, 2015.

[127] George Stefanou. The stochastic finite element method: past, present and future. *Computer Methods in Applied Mechanics and Engineering*, 198(9-12): 1031–1051, 2009.

[128] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[129] James William Thomas. *Numerical partial differential equations: conservation laws and elliptic equations*, volume 33. Springer Science & Business Media, 2013.

[130] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2):26–31, 2012.

[131] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. SIAM, 1997.

[132] Sumeet Trehan and Louis J Durlofsky. Trajectory piecewise quadratic reduced-order model for subsurface flow, with application to pde-constrained optimization. *Journal of Computational Physics*, 326:446–473, 2016.

[133] Jorn FM Van Doren, Renato Markovinović, and Jan D Jansen. Reduced-order optimal control of water flooding using proper orthogonal decomposition. *Computational Geosciences*, 10(1):137–158, 2006.

[134] PTM Vermeulen, Arnold W Heemink, and Chris BM Te Stroet. Reduced models for linear groundwater flow models using empirical orthogonal functions. *Advances in water resources*, 27(1):57–69, 2004.

[135] PTM Vermeulen, Chris BM Te Stroet, and Arnold W Heemink. Model inversion of transient nonlinear groundwater flow models using model reduction. *Water resources research*, 42(9):W09417, 2006.

[136] Zheng Wang, Dunhui Xiao, Fangxin Fang, Rajesh Govindan, Christopher C Pain, and Yike Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2017.

[137] Zhu Wang, Imran Akhtar, Jeff Borggaard, and Traian Iliescu. Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237:10–26, 2012.

[138] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(Issue 10):1550–1560, 1990.

[139] Andrew Westhead. *Upscaling for two-phase flows in porous media*. PhD thesis, California Institute of Technology, 2005.

[140] Karen Willcox. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers & fluids*, 35(2):208–226, 2006.

[141] M Winter and C Breitsamter. *Reduced-order modeling of unsteady aerodynamic loads using radial basis function neural networks*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV, 2014.

[142] D Xiao, Fangxin Fang, Andrew G Buchan, Christopher C Pain, Ionel Michael Navon, Juan Du, and G Hu. Non-linear model reduction for the navier–stokes equations using residual deim method. *Journal of Computational Physics*, 263: 1–18, 2014.

[143] Dunhui Xiao, Zhi Lin, Fangxin Fang, Christopher C Pain, Ionel M Navon, Pablo Salinas, and Ann Muggeridge. Non-intrusive reduced-order modeling for multiphase porous media flows using smolyak sparse grids. *International Journal for Numerical Methods in Fluids*, 83(2):205–219, 2017.

[144] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24 (2):619–644, 2002.

[145] Burak Yeten, Alexandre Castellini, Baris Guyaguler, and WH Chen. A comparison study on experimental design and response surface methodologies. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2005.

[146] Seonkyoo Yoon, Zeid Alghareeb, and John Williams. Development of reduced-order oil reservoir models using localized deim. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2014.

[147] Q Zhang, R Chassagne, and C MacBeth. Seismic history matching uncertainty with weighted objective functions. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, 2018.

[148] Daguang Zheng, Karlene A Hoo, and Michael J Piovoso. Low-order model identification of distributed parameter systems by a combination of singular value decomposition and the karhunen- loève expansion. *Industrial & Engineering Chemistry Research*, 41(6):1545–1556, 2002.

[149] Yuxiang Beckett Zhou. *Model reduction for nonlinear dynamical systems with parametric uncertainties*. PhD thesis, Massachusetts Institute of Technology, 2012.

[150] Hans.G Zimmermann, Christoph Tietz, and Ralph Grothmann. Forecasting with recurrent neural networks: 12 tricks. In *Neural Networks: Tricks of the Trade*, pages 687–707. Springer, 2012.